НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І

СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.005

«До захисту допущено»
Завідувач кафедри СПСКС
_____ В. П. Тарасенко
«___»_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

на тему: «СТРУКТУРНИЙ МЕТОД СИНТЕЗУ ДЖЕРЕЛА ПОСЛІДОВНОСТІ ТЕСТОВИХ ВЕКТОРІВ»

Виконав: студент ІІ курсу, групи КВ-71мп
Манілевич Дмитро Федорович

Науковий керівник:
к.т.н., доц. Потапова К.Р.

_____

_____

Рецензент: _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) – 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ В. П. Тарасенко
«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Манілевичу Дмитру Федоровичу

1. Тема дисертації «Структурний метод синтезу джерела послідовності тестових векторів», науковий керівник дисертації Потапова Катерина Романівна, к.т.н., доцент, затверджені наказом по університету від «30» жовтня 2018 р. № 4030-с

2. Термін подання студентом дисертації «7» грудня 2018 р.

3. Об'єкт дослідження: є методи структурної генерації тестових послідовностей двійкових векторів з постійною вагою.

4. Предмет дослідження: є процес синтезу джерела послідовності двійкових тестових векторів з постійною вагою, моделювання його функціонування.

5. Перелік завдань, які потрібно розробити:
- провести аналіз методів генерації послідовностей двійкових векторів;
- дослідити використання двійкових векторів з постійною вагою в якості тестових наборів;
- розробити метод генерації тестових послідовностей двійкових векторів;
- розробити програмне забезпечення для демонстрації роботи методу;
- проаналізувати результати роботи моделі.

6. Перелік ілюстративного матеріалу: презентація.

7. Перелік публікацій:
- Тези доповіді «Прикладна математика та комп'ютинг» ПМК-2018-2
- Тези доповіді на 20-й Міжнародній науково-технічній конференції SAIT 2018

8. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	15.10.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	07.03.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.04.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	11.05.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	08.06.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	13.07.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018	17.10.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	09.11.2018	

Студент Д. Ф. Манілевич

Науковий керівник дисертації К. Р. Потапова

РЕФЕРАТ

Актуальність теми.

Без цифрових систем зараз важко собі уявити розвиток людства. Вони знаходять використання майже в будь-якій галузі: від смартфона в кишені до управління величезним лайнером. Від їх справної роботи можуть залежать як майно, так і життя багатьох людей. Враховуючи це, важливою частиною розробки таких систем стає розрахунок їх надійності.

Основним способом тестування цифрових систем є генерування спеціальних впливів на об’єкт та аналіз його реакції. Найбільшого розповсюдження здобули послідовності псевдовипадкових сигналів. Одним із основних способів отримання таких послідовностей є генератори псевдовипадкових послідовностей, що використовують цифровий спосіб формування послідовностей. Генератори таких послідовностей мають ряд переваг над фізичними генераторами випадкових чисел.

При цьому важливою задачею є оптимізація тестування. Цифрові системи складаються з багатьох різних елементів, тому змога настроювати розподіл тестових наборів у відповідності до ймовірностей відмов елементів системи може призвести до оптимізації процесу тестування.

Об’єктом дослідження є методи структурної генерації послідовностей двійкових векторів.

Предметом дослідження є процес синтезу джерела послідовності двійкових векторів, моделювання його функціонування.

Мета роботи полягає в розробці методу синтезу джерела послідовності двійкових векторів постійної ваги з розподілом одиничних значень у вихідних векторах, що відповідає заданим ймовірнісним характеристикам.

Методи дослідження. В роботі використовуються методи проектування джерел псевдовипадкових двійкових чисел, методи теорії ймовірності, методи дискретної математики.

Наукова новизна роботи полягає в наступному:

Розроблено новий метод синтезу керованого генератора послідовності двійкових векторів з постійною вагою з розподілом одиничних значень у вихідних векторах, що відповідає вхідним ймовірнісним параметрам.

Практична цінність отриманих в роботі результатів полягає в використанні отриманого генератора для оптимізації процесу проведення статистичних тестів з моделями неоднорідних цифрових систем.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на XI науковій конференції молодих вчених «Прикладна математика та комп'ютинг» ПМК-2018-2 (Київ, 14-16 листопада 2018 р.) та на 20-й Міжнародній науково-технічній конференції SAIT 2018 (Київ, 21-24 травня 2018 р.)

Структура та обсяг роботи. Магістерська дисертація складається з вступу, трьох розділів, висновків та додатків.

У вступі надано загальну характеристику роботи.

У першому розділі виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету дослідження, розглянуто різноманітні методи отримання псевдовипадкових чисел.

У другому розділі розглянуто особливості використання псевдовипадкових послідовностей для формування тестових наборів, генератори псевдовипадкових тестових наборів, тестові послідовності спеціального виду, їх генерація та використання у статистичних тестах з моделями цифрових систем, основні аспекти нового методу.

У третьому розділі містяться основні положення нового методу, опис програмного продукту, аналіз роботи програмної моделі.

У висновках представлені загальні висновки та результати аналізу роботи моделі.

Робота виконана на 86 аркушах, містить __ додатків та посилання на список використаних літературних джерел з 9 найменувань. У роботі наведено 28 рисунків та 4 таблиці.

Ключові слова: неоднорідна цифрова система, псевдовипадковий двійковий вектор, постійна вага, регістр зсуву, лінійний зворотній зв’язок.

ABSTRACT

Theme urgency.

Without digital systems, it's hard to imagine the development of mankind. They find use in almost any industry: from a smartphone in a pocket to managing a huge liner. Both property and lives of many people relies on their correct work. Given this, an important part of the development of such systems is the calculation of their reliability.

The main way of digital systems testing is to generate special effects on the object and analyze its response. The most widespread are sequences of pseudorandom signals. One of the main ways of obtaining such sequences is generators of pseudorandom sequences that use the digital method of forming sequences. Generators of such sequences have several advantages over physical generators of random numbers.

At the same time, an important task is to optimize testing process. Digital systems consist of many different elements, so the ability to configure the distribution of test patterns in accordance with the probabilities of failure of the system elements can lead to optimization of the testing process.

Object of research are methods of sequences of binary vectors structural generating.

Subject of research is a process of synthesizing the source of the sequence of binary vectors, modeling its operation.

Research objective: is to develop a method for synthesizing a source of a sequence of binary vectors of constant weight with the distribution of ‘ones’ in output vectors, which corresponds to given probabilistic characteristics

Research methods. Methods of designing sources of pseudorandom binary numbers, theory of probability methods, discrete mathematics methods are used in this work.

Scientific novelty consists in the following:

A new method for synthesizing a controlled generator of a sequence of binary vectors with constant weight with the distribution of ‘ones‘ in output vectors, which corresponds to the input probabilistic parameters, is developed.

Practical value of the obtained results consists in use of the obtained generator to optimize the process of conducting statistical tests with models of heterogeneous digital systems..

Approbation. The basic points and outcomes of the research have been presented and discussed at the 11^{th} scientific conference for students and postgraduates «Applied mathematics and computing» PMK-2018 (Kyiv,

November 14-16, 2018) as well as at the 20th International Conference SAIT (May 21-24, 2018, Kyiv, Ukraine).

Structure and content of the thesis. The master thesis consists of the introduction, three chapters, conclusions and appendixes.

The introduction presents the general description of the research.

In the first chapter the estimation of the current state of the problem was made, the relevance of the research direction was substantiated, the purpose of the research was formulated, various methods of obtaining pseudorandom numbers were considered.

In the second chapter features of pseudorandom sequence use for formation of test patterns, generators of pseudorandom test patterns, test sequences of the special kind, their generation and use in statistical tests with digital systems models, main aspects of the new method were considered.

In the third chapter contains the main theses of the new method, description of the software product, analysis of the work of the software model are presented.

In the conclusions general conclusions and results of analysis of model work are presented.

The thesis is presented in 86 pages, it contains _ appendixes and 9 references to the used information sources. 28 figures and 4 tables are given in the thesis.

Key words: heterogeneous digital system, pseudorandom binary pattern, constant weight, shift register, linear feedback.

РЕФЕРАТ

Актуальность темы. Без цифровых систем сейчас трудно себе представить развитие человечества. Они находят применение почти в любой отрасли: от смартфона в кармане к управлению огромным лайнером. От их исправной работы могут зависят как имущество, так и жизнь многих людей. Учитывая это, важной частью разработки таких систем становится расчет их надежности.

Основным способом тестирования цифровых систем является генерирование специальных воздействий на объект и анализ его реакции. Наибольшее распространение получили последовательности псевдослучайных сигналов. Одним из основных способов получения таких последовательностей являются генераторы псевдослучайных последовательностей, использующих цифровой способ формирования последовательностей. Генераторы таких последовательностей имеют ряд преимуществ перед физическими генераторами случайных чисел.

При этом важной задачей является оптимизация тестирования. Цифровые системы состоят из многих различных элементов, поэтому возможность настраивать распределение тестовых наборов в соответствии с вероятностей отказов элементов системы может привести к оптимизации процесса тестирования.

Объектом исследования являются методы структурной генерации последовательностей двоичных векторов.

Предметом исследования является процесс синтеза источника последовательности двоичных векторов, моделирование его функционирования.

Цель работы заключается в разработке метода синтеза источника последовательности двоичных векторов постоянного веса с распределением единичных значений в сгенерированных векторах, что соответствует заданным вероятностным характеристикам.

Методы исследования. В работе используются методы проектирования источников псевдослучайных двоичных чисел, методы теории вероятности, методы дискретной математики.

Научная новизна работы состоит в следующем:

Разработан новый метод синтеза управляемого генератора последовательности двоичных векторов с постоянным весом с распределением единичных значений в исходных векторов, что соответствует входным вероятностным параметрам.

Практическая ценность полученных в работе результатов заключается в использовании полученного генератора для оптимизации процесса проведения статистических тестов с моделями неоднородных цифровых систем.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на XI научной конференции молодых ученых «Прикладная математика и компьютинг» ПМК-2018-2 (Киев, 14-16 ноября 2018) и на 20-й Международной научно-технической конференции SAIT 2018 (Киев, 21-24 мая 2018)

Структура и объем работы. Магистерская диссертация состоит из введения, трех разделов, выводов и приложений.

Во вступлении предоставлено общую характеристику работы.

В первом разделе выполнена оценка современного состояния проблемы, обоснована актуальность направления исследований, сформулированы цель исследования, рассмотрены различные методы получения псевдослучайных чисел.

Во втором разделе рассмотрены особенности использования псевдослучайных последовательностей для формирования тестовых наборов, генераторы псевдослучайных тестовых наборов, тестовые последовательности специального вида, их генерация и использование в статистических тестах с моделями цифровых систем, основные аспекты нового метода.

В третьем разделе содержатся основные положения нового метода, описание программного продукта, анализ работы программной модели.

В выводах сделаны общие выводы по работе; проанализированы полученные результаты.

Работа представлена на 86 страницах, содержит __ приложений и ссылки на список использованных литературных источников из 9 наименований. В работе приведены 28 рисунков и 4 таблици.

Ключевые слова: неоднородная цифровая система, псевдослучайный двоичный вектор, постоянный вес, регистр сдвига, линейная обратная связь.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АГТП	Автоматична генерація тестових послідовностей
БС	Багатопроцесорна система
ГПВП	Генератор псевдовипадкових послідовностей
ГПВЧ	Генератор псевдовипадкових чисел
ПВТН	Псевдовипадковий тестовий набір
ПВЧ	Псевдовипадкове число
РЗЛЗЗ	Регістр зсуву з лінійним зворотним зв’язком
РС	Регістр зсуву
ТН	Тестовий набір
ЦВ	Цифровий вузол
ЧП	Числова послідовність

ВСТУП

Без цифрових систем зараз важко собі уявити розвиток людства. Вони знаходять використання майже в будь-якій галузі: від смартфона в кишені до управління величезним лайнером. Від їх справної роботи можуть залежать як майно, так і життя багатьох людей. Враховуючи це, важливою частиною розробки таких систем стає розрахунок їх надійності.

Основним способом тестування цифрових систем є генерування спеціальних впливів на об’єкт та аналіз його реакції. Найбільшого розповсюдження здобули послідовності псевдовипадкових сигналів. Одним із основних способів отримання таких послідовностей є генератори псевдовипадкових послідовностей, що використовують цифровий спосіб формування послідовностей. Генератори таких послідовностей мають ряд переваг над фізичними генераторами випадкових чисел.

При цьому важливою задачею є оптимізація тестування. Цифрові системи складаються з багатьох різних елементів, тому змога настроювати розподіл тестових наборів у відповідності до ймовірностей відмов елементів системи може призвести до оптимізації процесу тестування.

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

1.1.1. Актуальність обраної теми

Для розрахунку ймовірності безвідмовної роботи багатопроцесорних систем на етапі проектування використовують статистичні тести над моделями, що відображають реакцію багатопроцесорної системи. Для таких тестів використовуються вектори станів системи $Z = \{z_1, z_2, \ldots, z_n\}$, де z_i – стан окремого елемента (процесора багатопроцесорної системи наприклад) і $z_i \in \{0,1\}$, де 0 – робочий стан, 1 – відмова.

Виходячи з цього, для оптимізації тестування таких систем, є доречним формування тестових послідовностей певної, постійної ваги k. Але якщо система неоднорідна, тобто ймовірності відмови різних елементів різні, то для подальшої оптимізації доречно використовувати послідовності постійної ваги, у яких на певних наборах одиничні значення з’являються частіше на позиціях тих елементів, які мають більшу ймовірність вийти з ладу.

1.1.2. Псевдовипадкові послідовності та їх генерація

1.2.1 Конгруентний метод

Одними з перших способів генерації псевдовипадкових послідовностей з рівномірним розподілом були конгруентні методи та методи генерації М-послідовностей.

Конгруентний метод описується рекурентним співвідношенням

$$X_k = AX_{k-1} + C(mod\ R), k = 1, 2, 3, \ldots, (1.1)$$

де A, C, R – постійні числа; $X_0 > 0$, $A > 0, C \geq 0, R > X_0, R > A, R > C$. При $C = 0$ метод матиме назву мультиплікативний конгруентний, а при $C > 0$ – змішаний конгруентний і послідовність, що формується у відповідності до нього називатиметься лінійною конгруентною послідовністю.

Найчастіше на практиці значення R вибирається як:

$$R = r^m,$$

де r – основа системи числення, що значно спрощує виконання операції по модулю числа R. Тоді алгоритм отримання чергового числа X_k матиме вигляд:

$$X_k = AX_{k-1} + C(mod\ r^m), k = 1, 2, 3, \dots . (1.2)$$

Проте не будь-який набір значень X_0, A, C, r, m призводить до формування послідовностей, близькими за своїми властивостями до рівномірним випадкових. Наприклад при $X_0 = A = C = 7$, $r = 10$, $m = 1$ числова послідовність, що формуватиметься матиме вигляд

$$\dots 6, 9, 0, 7\ , 6, 9, 0,\ 7, \dots .$$

У зв'язку з цим важливим є знаходження вимог до значень $X_0, A, C,\ r,$ m, при яких отримувана числова послідовність задовольняла хоча б даному мінімальному набору критеріїв[1]: рівноймовірність цифр всередині розрядів чисел послідовності; відсутність кореляції в розрядах та між ними.

Виходячи з теореми, що послідовність значень і-го розряду псевдовипадкових чисел, отриманих на основі виразу (1.2) при $C = 0$, $X_0 = 1$, є періодичною з періодом

$$L_i \leq r^{i-1}(r-1), i = 1, 2, \dots, m\ .$$

Можна побачити, що період послідовності значень розрядів псевдовипадкових чисел скорочується зі зменшенням номеру розряду. Так, для двійкової системи числення (r = 2), де $L_i \leq 2^{i-1}$, значення молодшої цифри псевдовипадкових чисел генерованої послідовності незмінно ($L_1 = 1$) , період значення другого розряду не перевищує двох, третього – чотирьох і т. д.

При $X_0 \neq 1, C = 0$ значення генерованої числової послідовності у відповідності до (2) визначається наступним чином:

$$X_1 = AX_0(mod\ r^m),$$
$$X_2 = A^2X_0(mod\ r^m),$$
$$\cdots$$
$$X_k = A^kX_0(mod\ r^m).$$

Отже, в даному випадку відбувається множення періодичної функції $F_k = A^k\ (mod\ r^m\ )$ на константу X_0. Оскільки таке множення не впливає на періодичні властивості,[2] то величина періоду значень і-го розряду псевдовипадкової числової послідовності буде також визначатися виразом

$$max\ L_i\ =\ r^{i-1}(r-1).$$

А беручи до уваги теорему, що послідовність значень і-го (і ϵ {1, 2, 3, …, m}) розряду псевдовипадкових чисел, що формуються у відповідності до (2) при $C \neq 0$, є періодичною з величиною інтервалу повторювання $T_i \leq r^{2i-1}(r-1)$, можна зробити висновок, що змішаний конгруентний алгоритм $(C \neq 0)$ забезпечує значно більший період повторення розрядних значень числової послідовності $\{X_k\}$ в порівнянні з мультиплікативним конгруентним методом ($C = 0$). В то самий час наявність в обох випадках періодичності всередині розрядів суттєво впливає на характеристики числової послідовності, що генерується. Характер даного впливу можна побачити на прикладі послідовності, що генерується у відповідності до виразу (1.2) при $C = 0, r = 2$ і значеннями $A,\ m, X_0$, при яких $L = 2^{m-1}$.

Математичне очікування $M[X_k]$ має вигляд:

$$M[X_k] = \frac{1}{2^{m-1}} \sum_{i=1}^{2^{m-1}} X_i = \frac{1}{2^{m-1}} \sum_{i=1}^{2^{m-1}} (2i-1) =$$
$$= 2^{m-1}. \quad (1.3)$$

При отримані виразу (1.3) враховувався той факт, що при $C = 0$ и $L_m = 2^{m-1}$ числова послідовність $\{X_k\}$ містить тільки непарні числа.

З урахуванням рівняння

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(n+2)}{6}$$

незміщена оцінка дисперсії послідовності $\{X_k\}$ визначається виразом

$$D[X_k] = \frac{2^{m-1}}{2^{m-1}-1}[\frac{1}{2^{m-1}}\sum_{i=1}^{2^{m-1}}(X_i^2 - M^2[X_k])] =$$

$$= \frac{2^{m-1}(2^{m-1}+1)}{3},$$

а функція коваріації

$$B_x[\tau] = \frac{2^{m-1}}{2^{m-1}-1}\left[\frac{1}{2^{m-1}}\sum_{i=1}^{2^{m-1}}(X_i X_{i+\tau} - \frac{2^{2m}}{4})\right].$$

З останнього виразу видно, що при $\tau = n * 2^{m-1}$, n = 1, 2, 3, … , значення $B_x[\tau]$ з точністю до множника дорівнює $D[X_k]$.

Для знаходження $B_x\tau$ при $\tau = n * 2^{m-2}$ виразимо X_i наступним чином:

$$X_i = [\,X_i - X_i(mod\ 2^{m-1})] + X_i(mod\ 2^{m-1}).$$

Вважаючи, що $X_i^0 = X_i - X_i^*$ і $X_i^* = X_i(mod\ 2^{m-1})$ є некорельованими випадковими величинами, то після певних математичних перетворень отримаємо:

$$B_x\left[\tau = n * 2^{m-1-i}\right] = \frac{2^{m-1}(2^{2m-2}-2^{2i})}{3*2^i(2^{m-1}-1)},$$

Або, переходячи з урахуванням дисперсії до автокореляційної функції

$$K_x\left[\tau = n * 2^{m-1-i}\right] = \frac{2^{2m-2}-2^{2i}}{2^i(2^{2m-2}-1)}$$

З отриманих результатів можна зробити наступні висновки: послідовність, що генерується на основі (1.2) при $R = r^m$, не задовольняє умові некорельованості значень розрядів псевдовипадкових чисел; результатом періодичності всередині розрядів слугує наявність додаткових сплесків автокореляційної функції на періоді повторення; недопустимо розбиття одного псевдовипадкового числа на окремі складові для отримання,

наприклад, координат багатовимірного вектору, так як якість компонент буде суттєво відрізнятись.

Існують різноманітні модифікації лінійного конгруентного методу, що дозволяють покращити ті чи інші характеристики генерованих числових послідовностей.

Наприклад квадратичний конгруентний метод

$$X_k = BX_{k-1}^2 + AX_{k-1} + C(mod\ R), k = 1, 2, 3, \dots$$

На практиці широке розповсюдження отримав варіант, що описується виразом

$$X_k = X_{k-1}(X_{k-1} + 1)(mod\ R), k > 0,$$

де X_0 вибирається із співвідношення $X_0(mo\ d\ 4) = 2; R = 2^m$. Даний алгоритм близький до методу середини квадрату фон Неймана, який є одним з перших способів генерування псевдовипадкових чисел[3].

Для збільшення періоду послідовності можна ввести додаткову залежність чергового числа X_k від більш ніж одного з попередніх значень. Один з найпростіших прикладів такого підходу реалізуються в алгоритмі формування послідовності Фібоначчі:

$$X_k = X_{k-1} + (X_{k-2})(mod\ \ R), \qquad k = 2, 3, 4, \dots$$

Лінійні конгруентні послідовності, що генеруються з використання алгоритму (1.2) або його модифікації, знаходять широкого застосування там, де потрібна рівномірність одновимірної густини розподілу ймовірностей. Проте при вирішені задач, що характеризуються підвищеними вимогами до багатовимірних розподілів, використання даних послідовностей є досить обмеженим. Причиною цього є n-вимірна неоднорідність розподілу ймовірностей, що особливо характерна для послідовностей малої довжини.

Ще одним недоліком конгруентного методу, що ускладнює його практичне використання, - суттєві обчислювальні затрати, пов’язані з необхідністю виконання в кожному такті генерування такої складної

операції як множення. Більш високою складністю обчислювань володіють різновиди алгоритму (1.2) при виборі значення R рівного $R = r^m - 1$, $R = r^m + 1$, чи рівного невеликому простому числу, що не перевищує r^m, що використовується для покращення періодичних властивостей послідовностей.

В силу даних недоліків лінійного конгруентного методу широкого розповсюдження набув спосіб формування псевдовипадкових послідовностей, що оснований на виразі

$$a_i = \sum_{k=1}^{m} \oplus \alpha_k a_{i-k}, i = 0, 1, 2, \ldots, (1.4)$$

де $\oplus$ - сума по модулю 2, і – номер такту, $a_i \in \{0,1\}$ – символи вихідної послідовності, $\alpha_k \in \{0,1\}$ – постійні коефіцієнти. При відповідному виборі коефіцієнтів $\{\alpha_k\}$ числова послідовність, що генерується має максимальну (для даного m) величину періоду і називається М-послідовністю.

1.2.2 М-послідовності

Однією з основних переваг методу генерування псевдовипадкових послідовностей максимальної довжини є простота його реалізації. Аппаратурний генератор М - послідовності, що функціонує у відповідності з (1.4), містить лише m-розрядний регістр зсуву і набір суматорів по модулю 2 в ланцюгу зворотного зв'язку. В процесі функціонування генератору регістр зсуву виконує зберігання і зсув вправо попередніх символів послідовності, а суматори в ланцюгу зворотного зв'язку виконують обчислення значень чергових символів, котрі послідовно записуються в самий лівий розряд регістру[4].

Якщо послідовність станів регістру зсуву представити як послідовність m-вимірних векторів $A = (a_1, a_2, \ldots, a_m)$, де $a_n \in \{0,1\}, n = \overline{1, m}$, то перетворення, виконувані схемою в деякому k-му такті роботи, можна записати у матричній формі:

$$\begin{vmatrix} a_1^{(k)} \\ a_2^{(k)} \\ a_3^{(k)} \\ \cdot \\ \cdot \\ \cdot \\ a_m^{(k)} \end{vmatrix} = \begin{vmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_{m-1} & \alpha_m \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ 0 & 0 & \dots & 1 & 0 \end{vmatrix} = \begin{vmatrix} a_1^{(k-1)} \\ a_2^{(k-1)} \\ a_3^{(k-1)} \\ \cdot \\ \cdot \\ \cdot \\ a_m^{(k-1)} \end{vmatrix} \quad (1.5)$$

Або в більш компактному вигляді $A^{(k)} = VA^{(k-1)}$, де

$$V = \begin{vmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_{m-1} & \alpha_m \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ 0 & 0 & \dots & 1 & 0 \end{vmatrix}.$$

Послідовне використання (1.5) дозволяє знайти стан регістру зсуву в довільний наступний такт роботи:

$A^{(k+s)} = V^s A^{(k)}$.

Циклічні властивості генератора послідовності визначаються характеристичним многочленом

$$\varphi(x) = \sum_{k=0}^{m} \oplus \alpha_k x^k ,$$

де $\alpha_0 = \alpha_m = 1$, $\alpha_j \in \{0,1\}, j = \overline{1, m-1}$. Останній представляє собою детермінант матриці $V \oplus xE$, що утворилася додавання змінноїx до діагональних елементів матриці V. Величина періоду послідовності, що генерується схемою, залежить від примітивності і незвідності $\varphi(x)$. Про те, що многочлен $\varphi(x)$ m −го порядку $(m = \deg \varphi(x))$ примітивний,

свідчить те, що поліном $x^k - 1$ ділиться на $\varphi(x)$ тільки при $k = 2^m - 1$. Незвідним називається такий многочлен $\varphi(x)$ степені m, який не ділиться ні на який інших многочлен від x нижчої степені. Послідовність максимальної довжини з періодом $2^m - 1$ формуватиметься тільки в тому випадку, коли характеризуючий (породжуючий) поліном $\varphi(x)$ примітивний і незвідний. При цьому необхідно, щоб початковий стан регістру зсуву не було нульовим, інакше послідовність, що генеруватиметься буде складатись з одних нулів, що відповідає нульовому – тривіальному циклу генератора.

Основна задача синтезу генератора псевдовипадкової послідовності максимальної довжини - знаходження многочлена $\varphi(x)$, що задовольняє умовам примітивності та незвідності. Відомо, що для даного $m = \deg(x)$ $\Phi(L = 2^m - 1)/m$ різних примітивних і незвідних поліномів, де $\Phi(L)$ – функція Ейлера. Оскільки з ростом m число $\Phi(L)$ швидко зростає, то відповідно зростає і кількість многочленів $\varphi(x)$ степені m, породжуючих М-послідовності. Серед цієї множини можна знайти поліноми, що мають найменше число ненульових членів. Даний випадок характерний для мінімальної конфігурації генератору псевдовипадкових чисел, де в склад ланцюгу зворотного зв'язку входить найменша кількість суматорів по модулю 2. Можна також зазначити, що для формування М-послідовностей разом з примітивним незвідним поліномом $\varphi(x)$ використовується і зворотній йому $\varphi^{-1}(x) = x_m\varphi(x^{-1})$.

Викладення питань теорії М-послідовностей зручно проводити з використання математичного апарату теорії скінченних алгебраїчних полів. Завдяки цьому зв'язку числові послідовності даного типу набули широкого використання в задачах завадостійкої передачі інформації.

Властивості послідовностей максимальної довжини:

1. Період М-послідовності, що формується у відповідності до виразу

$$a_i = \sum_{k=1}^{m} \oplus \alpha_k \, a_{i-k}, i = 0, 1, 2, \ldots,$$

дорівнює $L = 2^m$.

2. Для заданого $\varphi(x)$ існує L різних М-послідовностей, зсунутих відносно один одного.

3. Число одиничних символів на періоді М-послідовності дорівнює 2^{m-1}, а нульових - $2^{m-1} - 1$. Ймовірності появи 1 і 0 визначаються виразами і при збільшені m досягають значень як завгодно близьких до ½.

$$p(a_i = 1) = \frac{2^{m-1}}{2^m - 1} = \frac{1}{2} + \frac{1}{(2^{m+1} - 2)}$$

$$p(a_i = 0) = \frac{2^{m-1} - 1}{2^m - 1} = \frac{1}{2} - \frac{1}{(2^{m+1} - 2)}$$

4. В псевдовипадковій послідовності максимальної довжини серії з одного символу(одиниці чи нуля) зустрічаються 2^{m-2} разів, з двох одиниць чи нулів 2^{m-3}і т. д. Серії з $m - 1$ нуля чи m одиниць зустрічається лише по одному разу. Порівнюючи вирази для оцінки ймовірності появи серій з l однакових символів для випадкових послідовностей з відповідною ймовірністю для М-послідовності, можна впевнитись в їх практичній еквівалентності.

5. Для кожного цілого $s(1 \leq s < L)$ існує таке ціле $r \neq s$ $(1 \leq r < L)$, що $\{a_i\} \oplus \{a_{i-s}\} = \{a_{i-r}\}$. Дану властивість часто називають властивістю зсуву і додавання.

6. Автокореляційна функція М-послідовності визначається виразом

$$R_a(\tau) = \begin{cases} 1 \text{ при } \tau = 0 \ (mod\ L), \\ -\frac{1}{L} \text{ при } \tau \neq 0 \ (mod\ L). \end{cases}$$

7. Серед L ненульових М-послідовностей, сформованих на основі породжуючого поліному $\varphi(x)$, існує одна, що має властивість $a_i = a_{2i}$, $i =$

$0, 1, 2, \ldots$ Послідовності такого типу отримали розповсюдження в задачах завадостійкого кодування інформації і називаються характеристичними.

8. Децимацією послідовності $\{a_i\}$ по індексу $q (q = 1, 2, 3, \ldots)$ називається формування нової послідовності $\{b_i\}$ з q-х елементів $\{a_i\}$, тобто $b_i = a_{kq}$. Якщо $\{b_i\}$ ненульова послідовність, то вона породжується поліномом $\psi(x)$, корені якого представляють собою q-і степені коренів вихідного многочлена $\varphi(x)$, і має період $L/(L, q)$, де (L, q) – найбільший спільний дільник L і q. При $L,q=1$ період $\{b_i\}$ дорівнює $L = 2^{m-1}$, де $m = \deg \varphi(x)$, и децимація називається власною або нормальною.

1.2.3 Генератори псевдовипадкових чисел

Поряд з бінарними послідовностями, символи яких можуть приймати лише два значення, наприклад 0 і 1, 1 і 1 і т. д., широке практичне застосування мають послідовності, що складаються з рівномірно розподілених багаторозрядних псевдовипадкових чисел. Прилади, що слугують для формування таких послідовностей називаються генераторами псевдовипадкових чисел. Одним з найбільш розповсюджених шляхів реалізації останніх є використання для формування багаторозрядних чисел генераторів М-послідовностей.

Вирішення задачі синтезу генератора r-розрядних псевдовипадкових чисел може бути паралельне включення r генераторів М-послідовностей, кожний з яких призначений для формування значень одного з розрядів чисел. При цьому період М-послідовностей, а відповідно і величина РЗ генераторів визначається необхідною довжиною багаторозрядної псевдовипадкової послідовності. Однак на практиці подібний підхід не отримав значного розповсюдження в основному внаслідок складності приладу і невисоких статистичних характеристик генерованих послідовностей ПВЧ через наявність взаємної кореляції між М-послідовностями.

Найбільш часто на практиці при побудові r-розрядних генераторів ПВЧ використовується так званий послідовний принцип формування псевдовипадкових чисел, що полягає в формуванні чергового значення з r символів, послідовно отриманих за допомогою генератора М-послідовності. Чергове двійкове число A_k утворюється на виходах r розрядів регістру зсуву через кожні $s \geq r$ тактів роботи. Дане співвідношення представляє собою умову статистичної незалежності суміжних чисел формованої послідовності [5]. Довільний член псевдовипадкової числової послідовності $\{A_k\} = A_0, A_1, A_2, \ldots, A_k, \ldots$ можна описати виразом

$$A_k = a_1(ks)a_1(ks-1)a_1(ks-2)\ldots a_1(ks-r+1)$$

де $a_1(ks)$ – вміст першого розряду регістру зсуву в ks-ний такт роботи генератора. Враховуючи, що послідовності $\{a_1 ks\},\{a_1 ks-1\},\ldots,\{a_1 ks-r+1\}, k=0,1,2,\ldots$, періодичні, легко показати періодичність $\{A_k\}$ з величиною, що дорівнює частці від ділення числа 2^m-1(m - розрядність РЗ генератора) на найбільший спільний дільник чисел 2^m-1 та$-(2^m-1, s)$. Якщо $(2^m-1, s) \neq 1$, то генератор буде формувати $(2^m-1, s)$ різних псевдовипадкових числових послідовностей $\{A_k\}$, вид і характеристики яких залежать від початкового стану РЗ. При виборі s взаємнопростим з величиною $L = 2^m-1$ період $\{A_k\}$ буде дорівнювати L, а її характеристики не залежать від початкового стану регістру зсуву. Таким чином, для правильного вибору величини s необхідно використати розклад числа $L = 2^m-1$ на прості співмножники.

Для прикладу можна розглянути побудову послідовного генератора псевдовипадкових чисел $r = 4$ на основі генератора М-послідовності при $\varphi = 1 \oplus x^2 \oplus x^5$. Так як $L = 2^5-1 = 31$ є числом Мерсена, то в якості значення параметру s, що визначає число зсувів, вибирається $s = 4$. Функціональна схема такого ГПВЧ представлена на рисунку1.1, а в таблиці

1.1 представлені послідовності станів елементів РЗ генератора псевдовипадкових чисел $\{A_k\}=\{a_1|4k|a_1|4k-1|a_1|ks-2|a_1|4k-3\}$при початкових умовах $a_1(0) = 1,\ a_i(0) = 0, i = \overline{2,5}$.

В основі іншого розповсюдженого принципу побудови ГПВЧ лежить ідея використання різних ділянок однієї і тієї ж М-послідовності для формування незалежних значень розрядів генерованих чисел. Перевага цього принципу полягає в можливості одночасного отримання різноманітних ділянок М-послідовності за допомогою одного генератора шляхом введення в його склад додаткових суматорів по модулю два. Сигнали на виходах останніх розглядаються як значення розрядів псевдовипадкових чисел. Проте така реалізація ГПВЧ, що називаються паралельними, для використовуваних на практиці величин m і r відзначається складністю.

Оскільки кожен з r додаткових суматорів по модулю два має в середньому $m/2$ входів, затрати обладнання на їх побудову можуть в декілька разів перевищувати затрати на власне генератор М-послідовностей. Використання методів, що дозволяють мінімізувати число входів суматорів по модулю два, пов’язано зі значним об’ємом обчислювань і не призводить до значних виграшів. Окрім того, генерування псевдовипадкової числової послідовності у відповідності до даного методу відрізняється від описаного вище послідовного формування, по суті, лише технічною реалізацією. Це можна побачити на наступному прикладі.

Одним з підходів для вирішення цієї проблеми є метод тестування, заснований на використанні генераторів тестів на основі РЗЛЗЗ і перетворювачів генеруються послідовностей, які з окремих тестових наборів формують безліч детермінованих тестів, що виявляють цільові несправності в певних вузлах схеми. Послідовні ГПВЧ володіють мінімальними апаратними затратами, недосяжними для паралельних пристроїв, але суттєво програють їм в швидкодії.

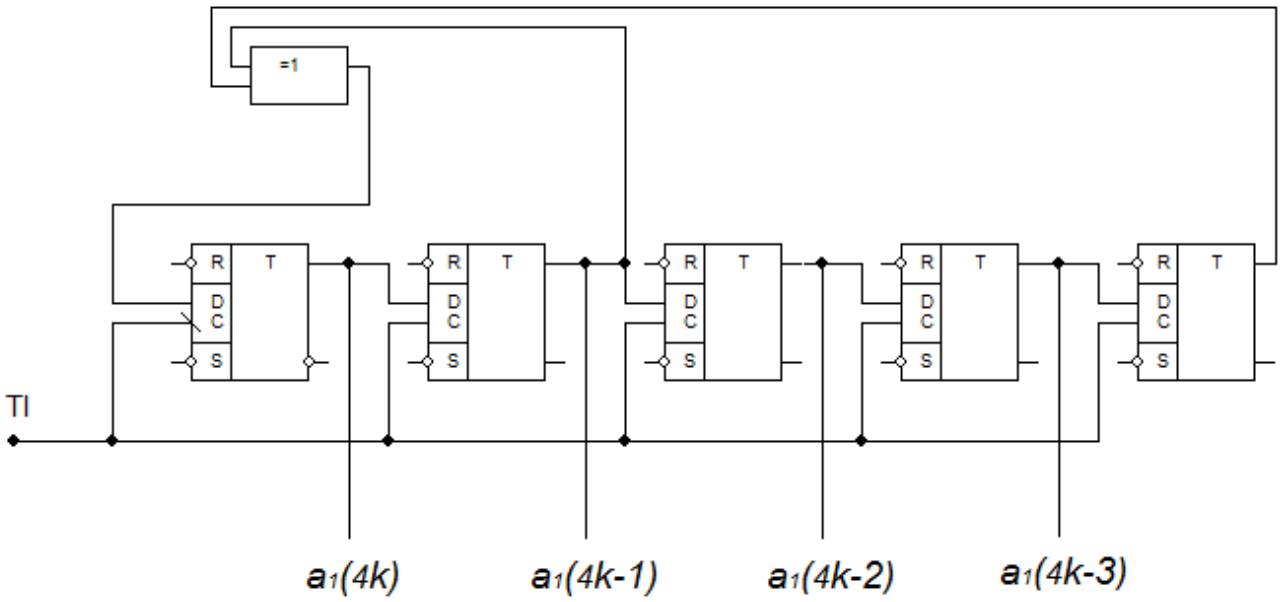


Рисунок 1.1- Чотирьохрозрядний послідовний ГПВЧ для $\varphi = 1 \oplus x^2 \oplus x^5$ та $s = 4$

Таблиця 1.1 - Послідовність станів елементів ГПВЧ і генерованих чисел

k	$a_1(k)a_2(k)a_3(k)\,a_4(k)a_5(k)$	$a_1(4k)a_1(4k-1)a_1(4k-2)\,a_1(k-3)$
0	1 0 0 0 0	1 0 0 0
1	0 1 0 0 0	
2	1 0 1 0 0	
3	0 1 0 1 0	
4	1 0 1 0 1	1 0 1 0
5	1 1 0 1 0	
6	1 1 1 0 1	
7	0 1 1 1 0	
	…	…
14	1 1 0 0 0	
15	1 1 1 0 0	

22	0 1 1 0 0	
23	1 0 1 1 0	

Нехай для породжуючого поліному $\varphi = 1 \oplus x^2 \oplus x^5$ необхідно побудувати 4-х розрядний ГПВЧ. При цьому значення першого розряду чисел послідовності визначається вмістом першого розряду $a_1(k)$ РЗ генератора М-послідовності, а другого, третього і четвертого розрядів – значеннями $a_1(k+23)$, $a_1(k+15)$ та $a_1(k+7)$ відповідно. Для побудови ГПВП необхідно знайти коефіцієнти $\{\delta_i(l)\}, i = \overline{1,5}$ для $l = 7, 15, 23$, де $\delta_i(l)$ = коефіцієнт, що визначає використання значення i-го розряду РЗ для формування елементу зсунутої послідовності. Для цього можна використати відому методику [5]. Вектори станів елементів РЗ генератора М-послідовності $a_1(k+6), a_2(k+6), a_3(k+6), a_4(k+6), a_5(k+6)$ = 11101, $a_1(k+14)$, $a_2(k+14)$, $a_3(k+14)$, $a_4(k+14), a_5(k+14)$ = 11000, $a_1(k+22)$, $a_2(k+22), a_3(k+22), a_4(k+22), a_5(k+22)$ = 01100 (див. таблицю 1.1) множаться на матрицю

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \\ \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & 0 \\ \alpha_3 & \alpha_4 & \alpha_5 & 0 & 0 \\ \alpha_4 & \alpha_5 & 0 & 0 & 0 \\ \alpha_5 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

В результаті будуть отримані значення коефіцієнтів $\{\delta_i(l)\}$ (таблиця 1. 2).

Функціональна схема ГПВП показана на рисунку 1.2, а в таблиці 1.3 представлена послідовність станів РЗ генератора і псевдовипадкових чисел $\{A_k\} = \{a_1(k), a_1(k+23), a_1(k+15), a_1(k+7)\}$, що формуються при початкових умовах $a_1(k), a_2(k), a_3(k), a_4(k), a_5(k)$ = 10101, k=0.

Порівнюючи результати, що містяться в таблиці 1.1 та 1.3 можна впевнитись, що послідовності псевдовипадкових чисел, що формуються послідовними і паралельними способами, є абсолютно тотожними. В той

самий час реалізація даних способів принципово відрізняється і має різну швидкодію і апаратні затрати. Послідовні ГПВЧ володіють мінімальними апаратними затратами, недосяжними для паралельних пристроїв, але суттєво програють їм в швидкодії.

Таблиця 1.2 - Значення коефіцієнтів $\{\delta_i(l)\}$

l	$\delta_1(l)$	$\delta_2(l)$	$\delta_3(l)$	$\delta_4(l)$	$\delta_5(l)$
7	0	1	1	1	1
15	1	1	0	1	1
23	1	0	1	1	0

Одним з способів підвищення швидкості формування ПВЧ послідовностей можна розглянути на конкретному прикладі генератора дев'ятирозрядних чисел, що формуються на основі М-послідовності, що породжується поліномом $\varphi = 1 \oplus x^4 \oplus x^9$ розглянутий в роботі (6). Дане рішення основується на реалізації наступного співвідношення, що описує роботу генератора М-послідовності:

$$A^{(k+s)} = V^s A^{(k)}, (1.6)$$

де $A^{(k+s)} = \big(a_1(k+s), a_2(k+s), \dots a_m(k+s)\big)$та $A^{(k)} = (a_1(k), a_2(k), \dots a_m(k))$ – m- вимірні вектори, що відображають стани РЗ генератора в $(k+s)$-ий та k- ий такти його роботи відповідно, V – матриця, що визначається значеннями коефіцієнтів

$$\{\alpha_1\}, i = \overline{1,m},$$

породжуючого поліному $\varphi(x)$, що описує структуру генератора. При $s = 1$ пристрій, що функіонує у відповідності до (1.6), є, по суті, звичайним генератором М-послідовності.

Для $s \geq 1$, згідно з (1.6), протягом одного такту буде відбуватися s-розрядний зсув М-послідовності. При цьому функціональні з'язки в генераторі будуть описуватися матрицею V^s.

Таблиця 1.3 - Стани регістру зсуву ГПВЧ і послідовності згенерованих чисел

k	$a_1(k)$	$a_2(k)$	$a_3(k)$	$a_4(k)$	$a_5(k)$	$A_k = a_1 k a_1 k+23$ $a_1(k+15)a_1(k+7)$
0	1	0	1	0	1	1 0 0 0
1	1	1	0	1	0	1 0 1 0
2	1	1	1	0	1	1 0 1 1
3	0	1	1	1	0	0 0 0 1
4	1	0	1	1	1	1 1 1 1
5	1	1	0	1	1	1 0 0 1
6	0	1	1	0	1	0 1 0 1

Для прикладу можна розглянути побудову послідовного $r = 4$-розрядного ГПВЧ для полінома $\varphi = 1 \oplus x^2 \oplus x^5$ і величини зсуву $s = 4$. матриця V^4 має вигляд

$$V^4 = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^4 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^4 =$$

$$= \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Тоді вираз (1.6) запишеться як

$$A^4 = \begin{bmatrix} a_1(k+4) \\ a_2(k+4) \\ a_3(k+4) \\ a_4(k+4) \\ a_5(k+4) \end{bmatrix}^4 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1(k) \\ a_2(k) \\ a_3(k) \\ a_4(k) \\ a_5(k) \end{bmatrix}. \quad (1.7)$$

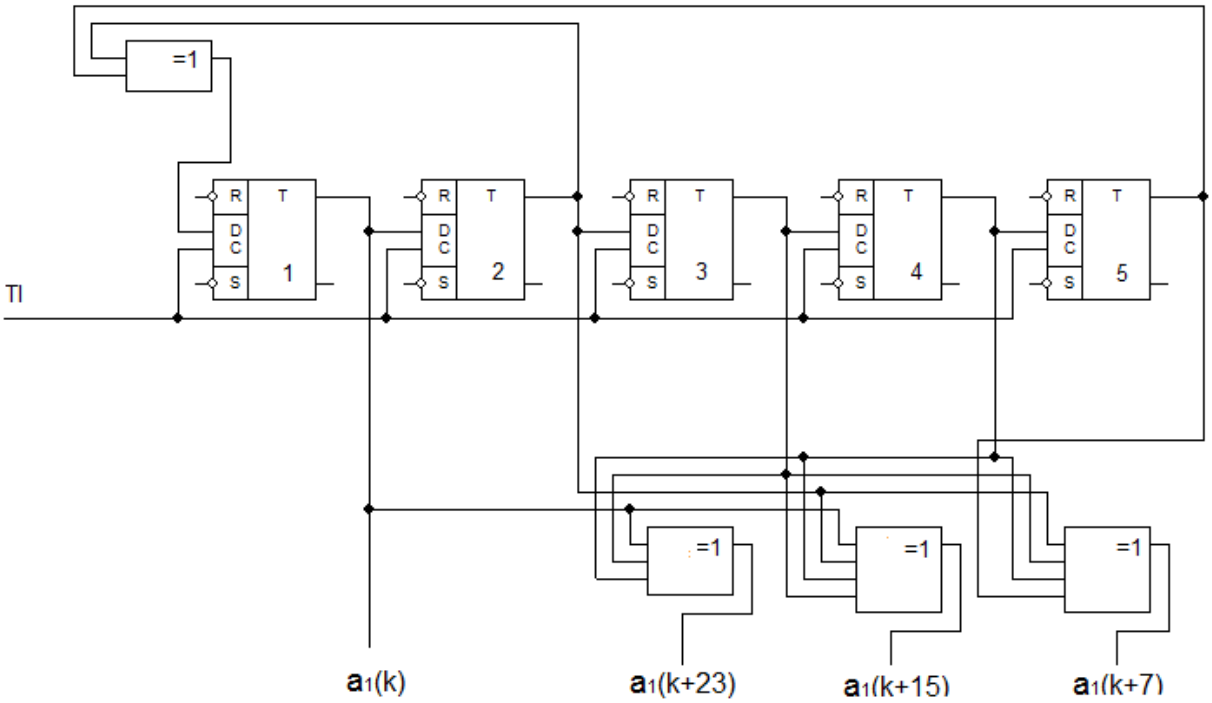


Рисунок 1.2 - Функціональна схема чотирьохрозрядного ГПВЧ для $\varphi = 1 \oplus x^2 \oplus x^5$

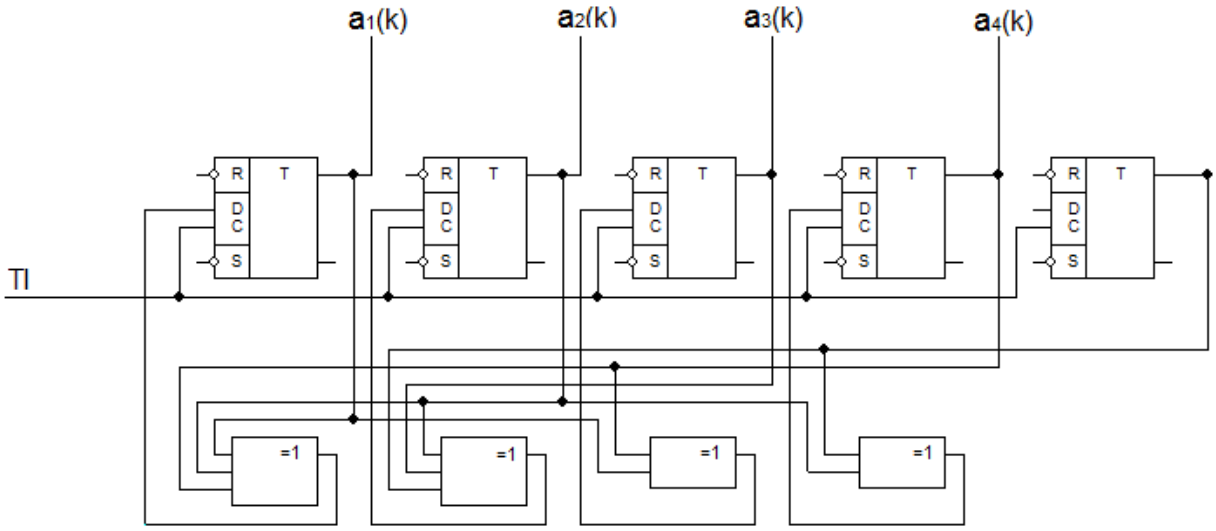
На основі (1.7) будується система логічних рівнянь, що визначає функціональну схему генератора:

$$a_1(k+4) = a_1(k) \oplus a_2(k) \oplus a_4(k),$$
$$a_2(k+4) = a_2(k) \oplus a_3(k) \oplus a_5(k),$$
$$a_3(k+4) = a_1(k) \oplus a_4(k),$$
$$a_4(k+4) = a_2(k) \oplus a_5(k),$$
$$a_5(k+4) = a_1(k).$$

Чергове число може псевдовипадкове число може визначатися, наприклад, як вектор $a_1(k)a_2(k)\,a_3(k)a_4(k)$ (рисунок 1.3). Послідовність станів запам'ятовуючих елементів даного ГПВЧ при початкових умовах $a_1(k) = 1, a_i(k) = 0, i = \overline{2,5}$ приведена в таблиці 1.4.

Легко переконатись, що послідовність псевдовипадкових чисел, що формується генераторами, схеми яких представлені на рисунку 1.1, 1.2 та 1.3, є тотожними.

Рисунок 1.3 - Функціональна схема четирьохрозрядного ГПВЧ для $\varphi =$



$1 \oplus x^2 \oplus x^5$ і $s = 4$ зі зв'язками, що описуються матрицею V^4

Порівнюючи структури послідовних ГПВЧ, що приведені на рисунку 1.1 та 1. 3, з точки забезпечення високої швидкодії, потрібно віддати останній. В той самий час з порівняння її з структурою паралельного генератора (рисунок 1. 2) слідує еквівалентність забезпечуваних ними апаратних витрат і швидкодії. При цьому М-послідовності, що використовуються в генераторах двох останніх типів для формування псевдовипадкових чисел, не завжди описуються одним і тим же примітивним поліномом $\varphi(x)$. Цей випадок трапляється тільки тоді, коли результатом децимації М-послідовності по індексу s є та сама М-послідовність, що породжується поліномом $\varphi(x)$. Також варто відмітити, що для ГПВЧ, що реалізує багаторозрядний зсув за один такт, завжди можна побудувати еквівалентну структуру паралельного генератора. В той же час зворотне твердження справедливо не завжди. Проте, якщо говорити про про оптимальну (в плані апаратних затрат) структуру паралельного ГПВЧ , то вона однозначно реалізується при організації багаторозрядного зсуву на базі поліномів з мінімальною кількістю ненульових коефіцієнтів.

Таким чином, співвідношення (1.6) є визначаючим для побудови практичних схем швидкодіючих ГПВЧ. При цьому апаратна складність генератора залежить від кількості ненульових елементів матриці V^s.

Більш загальний підхід до побудови ефективних багаторозрядних ГПВЧ показано в [8]. Якщо примітивний незвідний поліном $\varphi(x)$ М-послідовності, що використовується при генерації псевдовипадкових чисел, представляється у вигляді

$$\varphi(x) = 1 \oplus \varphi_1(x) \oplus \varphi_2(x) \ldots \oplus \varphi_d(x), \quad (1.8)$$

то ГПВЧ може бути реалізований на базі генераторів М-послідовностей меншої розрядності, ніж $m = \deg(\varphi(x))$, породжуючі поліноми яких можуть бути представлені у вигляді (1.7).

Зокрема, якщо породжуючий поліном має вигляд $\varphi(x) = 1 \oplus x\left(1 \oplus x^{m-j}\right)$, що справедливо для будь-якого многочлена $\varphi(x) = 1 \oplus x^j \oplus x^m$, то функціональна схема ГПВЧ включає послідовно з'єднані в кільцеву схему $(m - j)$ D-тригерів і j Т-тригерів.

У випадку, коли породжуючий поліном описується виразом $\varphi(x) = 1 \oplus (1 \oplus x)^j\left(1 \oplus (1 \oplus x)^{m-j}\right)$, ГПВЧ складається з m послідовно з'єднаних Т-тригерів, до того ж на вхід першого тригера подається інформація із суматора по модулю два, підключеного до виходів m-го та $m-j$-го тригерів.

Декілька більш сучасних прикладів генераторів, що також використовують регістри зсуву з лінійним зворотнім зв'язком . Наприклад Каскад Голлманна чи Stop-and-Go генератор.

Каскад Голлмана складається з серії РЗЛЗЗ(LFSR- linear feedback shift register), при чому такт кожного наступного LSFR контролюється попереднім

(тобто якщо вихід LFSR-1 дорівнює 1 в момент часу $t-1$, то LFSR-2 змінює своє значення на наступне). Вихід послідовності LFSR є виходом всього генератора. Якщо всі LFSR – довжини l, то лінійна складність системи с n

LFSR дорівнює $l*(2^l-1)^{n-1}$.

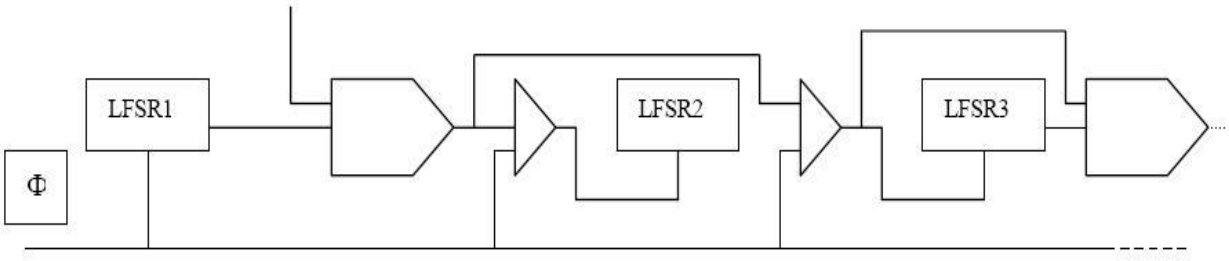


Рисунок 1.4 - Каскад Голлмана

Stop-and-Go генератор. В цьому генераторі використовується 3 LFSR різної довжини. LFSR-2 змінює свій стан, якщо вихід LFSR-1 дорівнює 1; LFSR-3 змінює свій стан в протилежному випадку. Результатом генератора есть XOR LFSR-2, LFSR-3. У такого генератора великий період і велика лінійна складність.

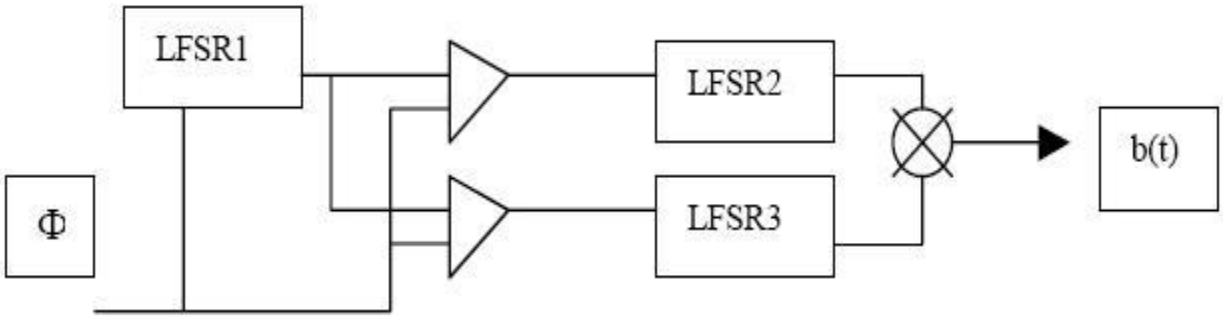


Рисунок 1.5 - Stop-and-Go генератор

1.2.4 Багатоканальні генератори псевдовипадкових послідовностей

При вирішені ряду практичних задач, пов’язаних с використання псевдовипадкових числових послідовностей (генерація сигналів збудження в системах автоматизації багатокомпонентних динамічних випробуваннях, перетворення даних в системах зв’язку з шумоподібними сигналами, тощо) , виникає необхідність одночасного формування декількох незалежних псевдовипадкових числових послідовностей. Очевидний шлях для досягнення даної мети – паралельне включення декількох генераторів,

характеризується значними апаратними витратами і потребує спеціальних заходів по забезпеченню декорельованості генерованих псевдовипадкових числових послідовностей. Більш високих показників можна досягти при використанні метода побудови швидкодіючих послідовних ГПВЧ, в основі яких лежить реалізація виразу (1.6). Дійсно, отримавши аналітичні співвідношення, що встановлюють залежності між векторами станів РЗ генератора в різні такти його роботи, а потім реалізувавши у співвідношенні з ним схему пристрою, отримують багатоканальний ГПВЧ послідовностей [5].

1.3 Висновки

Генерація псевдовипадкових послідовностей є важливою задачею з надзвичайно різноманітними способами використання. Среде різним математичних принципів для формування псевдовипадкових чисел найбільшого розповсюдження набули так званий конгруентний метод та метод М-послідовностей. Викладення питань теорії М-послідовностей зручно проводити з використання математичного апарату теорії скінченних алгебраїчних полів. Завдяки цьому зв’язку числові послідовності даного типу набули широкого використання в задачах завадостійкої передачі інформації. Генератори, що будуються на цих принципах також можуть сильно відрізнятися структурно (послідовні, паралельні), за швидкодією та апаратними витратами. Значної популярності здобули генератори на основі регістра з зсувом.

ТЕСТУВАННЯ РІЗНОМАНІТНИХ ЦИФРОВИХ СИСТЕМ, ФОРМУВАННЯ ПСЕВДОВИПАДКОВИХ ДВІЙКОВИХ ТЕСТОВИХ ВЕКТОРІВ

2.1 Особливості використання псевдовипадкових послідовностей для формування тестових наборів

В процесі виробництва та експлуатації засобів обчислювальної техніки виникає задача контролю та діагностики цифрових вузлів. Значимість її постійно зростає з ускладненням конструкції самих вузлів і підвищення ступеня інтеграції використовуваних в них компонентів. Це і стало причиною інтенсивних розробок в області методів і засобів автоматизованого тестового контролю і діагностики цифрових вузлів. В найбільш загальному вигляді принцип дії систем такого призначення полягає в подачі на досліджуваний об'єкт послідовності тестових наборів, генерованих апаратно чи програмно, и аналізі сигналів на його виході с метою виявлення несправностей чи збоїв.

Повнота контролю ЦВ при використанні випадкових чисел в якості тестових наборів може бути оцінена величиною $P_0\,(N)$[3], що дорівнює ймовірності виявлення несправностей і визначається як відношення виявлених несправностей до загального числа несправностей ЦВ при використанні N тестових наборів, що представляють собою випадкові числа. Ймовірність $P_0(N)$ можна виразити як

$$P_0(N) = 1 - P_1(N),$$

де $P_1(N)$ – спадна с ростом N функція ймовірності не виявити несправність.

При $N = 0$ $P_1\,(N) = 1,$ а при $N \to \infty$ $limP_1\,(N) = P_2,$ де $P_2 = const$ визначається видом тестових наборів, що використовуються і перевіряємого ЦВ і обчислюється як відношення невиявлених несправностей до їх загального числа. Таким чином, очевидні вимоги до ТН, що використовуються при контролі ЦВ: забезпечення рівності $P_2 = 0$ і швидкої збіжності $P_1(N)$ до P_2з ростом N. Тільки в такому випадку за

прийнятний час, що визначається величиною N, буде забезпечена максимальна повнота контролю[7].

З точки зору виконання вимоги $P_2 = 0$ доцільно використовувати генератори тестових наборів, що основуються на фізичних принципах . Однак необхідність забезпечення максимальної швидкодії і розрядності генератору ТН, а також відтворення дослідних сигналів практично не дозволяє використовувати такі пристрої в системах тестового контролю.

У зв'язку з цим, на практиці розповсюджені апаратно генеровані псевдовипадкові тестові набори великої довжини. Вони використовуються в системах контролю, що основуються на методах випадкового пошуку, сигнатурного аналізу, реалізованих на ймовірнісних принципах, а також при організації самотестування великих і надвеликих інтегральних схем. При цьому в якості псевдо випадкових тестових наборів в більшості випадків використовуються М-послідовності. Однак такий підхід обмежує кількість цифрових вузлів, що можна контролювати. Причиною цього є детермінованість структури даних псевдовипадкових тестових наборів, яка виражається в залежності між її символами.

2.2 Генератори псевдовипадкових тестових наборів

Генератори псевдовипадкових тестових наборів, які дозволяють забезпечити виконання $P_2 = 0$ при перевірці ЦВ, що представляє собою синхронну послідовну схему, мають відповідати певним вимогам[7].

Будь-яка послідовна схема може бути представлена так званою моделлю Хаффмена, для якої $A(k) = a_1(k), \dots a_l(k) \dots a_L(k)$ та $Y(k) = y_i(k), \dots, y_r(k), \dots, y_R(k)$ є відповідно вхідними і вихідними змінними в k-ий такт її функціонування, а $Z(k) = z_1(k), \dots z_q(k) \dots z_Q(k)$ – внутрішніми змінними, що визначають стан схеми.

Функціонування схеми описується системою рівнянь

$$y_r(k) = f_r\left[a_1(k), \dots a_l(k) \dots a_L(k), \quad z_1(k), \dots z_q(k) \dots z_Q(k)\right] = \\ = f_r[A(k), Z(k)], \; r = \overline{1,R}, \quad (2,1)$$

$$z_q(k+1) = \psi_q\big[(k), \dots a_l(k) \dots a_L(k), \quad z_1(k), \dots z_q(k) \dots z_Q(k)\big] =$$
$$= \psi_q[A(k), \quad Z(k)], q = \overline{1,Q}, \quad k = 0,1,2,\dots$$

Припустивши, що в схемі відсутні кільця зворотного зв'язку і використовуються тільки синхронні елементи пам'яті, що виконують лише функції затримки, її можна розбити на декілька рівнів послідовно функціонально зв'язаних елементів пам'яті. При цьому математична модель такої послідовної схеми, отриманої з системи рівнянь (2.1), представлена в векторній формі, буде мати вигляд

$$Z_1(k+1) = \psi[A(k)],$$
$$Z_2(k+1) = \psi[A(k), Z_1(k)],$$
$$Z_3(k+1) = \psi[A(k), Z_1(k), Z_2(k)],$$
$$\dots\dots\dots$$
$$Z_c(k+1) = \psi[A(k), Z_1(k), Z_2(k), \dots, Z_{c-1}(k)],$$
$$Y(k) = f[A(k), Z(k), Z_2(k), \dots, Z_c(k)\ ],$$

де $Z_1(k) \cup Z_2(k) \cup Z_3(k) \cup \dots \cup Z_c(k) = Z(k)$.

Таким чином, послідовна схема представляється у вигляді $C \leq Q$ рівнів функціонально зв'язаних елементів пам'яті, при чому стан певного елемента пам'яті i-го рівня $(i=\overline{1,c})$ в $(k+1)$- й такт роботи схеми залежить тільки від станів елементів пам'яті попередніх рівнів в k-й такт і значень вхідних змінних $A(k)$.

Отриману систему рівнянь можна перетворити до вигляду

$$Z_1(k+1) = \psi[A(k)],$$
$$Z_2(k+1) = \psi[A(k), A(k-1)],$$
$$Z_3(k+1) = \psi[A(k), A(k-1), A(k-2)],$$
$$\dots\dots\dots \qquad (2.2)$$
$$Z_c(k+1) = \psi[A(k), A(k-1), A(k-2), \dots, A(k-c+1)],$$
$$Y(k) = f[A(k), A(k-1), A(k-2), \dots, A(k-c)\ ].$$

На основі (2.2) система рівнянь (2.1) в векторній формі матиме вигляд

$$Z(k) = \psi[A(k), A(k-1), A(k-2), \dots, A(k-c+1)],$$
$$Y(k) = f[A(k), A(k-1), A(k-2), \dots, A(k-c)\ ].$$

Використовуючи дані вирази, можна показати, що значення перемикаючою функції F_j на j-й шині послідовної схеми (під шиною схеми, що перевіряється, розуміють вихід схеми, вихід елементу пам'яті чи вихід комбінаційного елемента, що входить в склад схеми) визначається як $F_j = F_j[A(k), A(k-1), A(k-2), \dots, A(k-c)]$.

При формуванні вхідних змінних $A(k)$ з використання ГПВТН можливий випадок, коли $F_j = const$. Виникнення шини, що не перевіряється, де $F_j = con\ st$, пояснюється тим, що при формуванні вхідних тестових наборів (в тому числі і при використанні ГПВТН) значення функції F_j перевіряється не на всіх можливих наборах $A(k), A(k-1), A(k-2), \dots, A(k-c)$. Набори, на яких перевіряється j- та шина ЦВ, складається з $(c+1)L$ двійкових символів. Тому, якщо між символами набору існує функціональний зв'язок, що призводить до виключення певних комбінацій в них, можливий випадок, коли $F_j = const$. При формуванні всіх можливих комбінацій з $(c+1)L$ двійкових символів в тестових наборах при перевірці j-ї шини справного ЦВ $F_j \neq const$. У випадку використання апаратних генераторів тестових наборів, побудованих за допомогою М-послідовностей, умова формування всіх можливих $(c+1)L$ –розрядних вхідних наборів. При цьому буде виконуватись умова $P_2 = 0$ для довільного ЦВ, що містить не більше ніж c послідовних функціонально зв'язаних рівнів елементів пам'яті.

Методика синтезу апаратних генераторів псевдовипадкових тестових наборів, що забезпечують максимальну повноту контролю, складається з наступних етапів:

Для заданої множини ЦВ визначається максимальна кількість послідовно з'єднаних елементів пам'яті c з максимальним числом входів L.

З відомих таблиць примітивних незвідних поліномів [4] вибирається такий поліном $\varphi(x)$, для якого виконується нерівність $\deg(\varphi(x)) > (c+1)L$.

По розглянутій вище методиці будується ГПВЧ, що реалізує L-розрядний зсув інформації. Синтезований таким чином ГПВЧ представляє собою ГПВТН, для якого забезпечується $(c+1)$-вимірний закон розподілу L-розрядних наборів.

Входами ГПВТН будуть виходи L будь-яких послідовних його розрядів. Таким чином, максимальна повнота контролю довільного цифрового вузла з використанням апаратних генераторів псевдовипадкових тестових послідовностей досягається при наступних основних умовах:

В ЦВ, що перевіряється, в процесі контролю відсутні ланцюги зворотного зв'язку.

Послідовність ПВТН має $(c+1)$-вимірний закон рівномірного розподілу.

Тільки в цьому випадку в процесі контролю буде виконуватися рівність $P_2 = 0$, що свідчить про відсутність фрагментів ЦВ, що не перевіряються.

Забезпечення рівності $P_2 = 0$ і відповідно $P_1(N) = 0$ при $N \to \infty$ є лише одним показником, що характеризує якість НВТН. Іншим показником слугує швидкість сходження $P_1(N)$ до нуля з ростом величини N, яка визначає довжину послідовності ТН. Тому в реальних автоматизованих системах контролю при формуванні НВТН вирішується задача збільшення швидкості сходження $P_1(N)$ до нуля і, таким чином, забезпечується прийнятна довжина тестової послідовності.n

Вбудоване самотестування (Built-in-Self-Test (BIST)) дискретних пристроїв (ДП) широко використовується у виробництві сучасних ДП на рівні друкованих плат і отримує подальший розвиток при проектуванні і створенні систем на одному кристалі (СОК). Підвищення швидкодії логічних елементів, складності СОК і необхідності перевірки їх справності

на робочих частотах ДП, що досягають 15 Ггц, визначають переваги BIST перед зовнішніми засобами діагностування. У роботах відомих фахівців провідних зарубіжних фірм і корпорацій - розробників і виробників СОК була запропонована ідея поєднання концепції BIST і стандарту проектування IEEE 1149.1 "Периферійне сканування". Цей стандарт визначає структуру СОК, в якій вхідні регістри функціонують в двох режимах: робочому і тестовому. У тестовому режимі вхідні регістри СОК реконфігуруются в регістри зсуву (РЗ), що дозволяє вводити і виводити діагностичну інформацію через стандартний порт JTAG і звести процедуру діагностування до перевірки справності комбінаційної частини ДУ. У режимі самотестування вхідні регістри реконфігуруются в генератори тестів і еталонних сигнатур, а вихідні регістри в синдромно-сигнатурні аналізатори вихідних реакцій ДП, що перевіряється. В якості генераторів тестів використовуються регістри зсуву з лінійними і нелінійними зворотними зв'язками (РЗЛЗЗ і РЗНЗЗ), мережі клітинних автоматів (МКА).

Для перевірки справності ДП використовуються різні методи компактного тестування: вичерпне, псевдовичерпне, псевдовипадкове тестування на основі РЗЛЗЗ, РЗНЗЗ і МКА, тестове діагностування детермінованим безліччю тестів, які 4 формуються на етапі проектування ДП програмними системами генерації тестів і моделювання несправностей. Ефективність вбудованих засобів діагностування ДП визначається повнотою виявлення несправностей, часом тестування і апаратними витратами на реалізацію цих коштів.

Як показує досвід впровадження BIST виробниками СОК, не більше 5-10% площі кристала допускається використовувати для реалізації вбудованих засобів діагностування. Тому розробка методів синтезу і проектування ефективних технічних засобів BIST, що відповідають перерахованим вище критеріям, є актуальною проблемою.

Широке поширення методів компактного тестування ДУ шляхом використання генераторів псевдовипадкових і псевдовичерпних тестів з

подальшим стисненням вихідний реакції синдромно-сигнатурними аналізаторами як вбудованих засобів діагностування СОК визначило ряд нових проблем при проектуванні СОК і їх діагностичного забезпечення. Перша проблема пов'язана з необхідністю підвищення ступеня покриття несправностей СОК щонайменше для класу константних несправностей, друга - обумовлена необхідністю зниження споживаної енергії в процесі тестового діагностування СОК.

Відомо, що довжина псевдовичерпних тестів для СОК з числом входів 50-200 може становити 10^4-10^6 тестових наборів. При цьому, як правило, не виявляються несправності "важко" керованих вузлів перевіряються схем. Крім того, так як споживана енергія в КМОН схемах пропорційна числу перемикань її елементів, то скорочення довжини тестових послідовностей дозволить скоротити час тестування і мінімізувати енергетичні витрати на технічне обслуговування СОК.

Одним з підходів для вирішення цієї проблеми є метод тестування, заснований на використанні генераторів тестів на основі РЗЛЗЗ і перетворювачів генеруються послідовностей, які з окремих тестових наборів формують безліч детермінованих тестів (ДТ), що виявляють цільові несправності в певних вузлах схеми (рисунок 2.1). При цьому довжина тестової послідовності не змінюється. Використання безлічі ДТ, що покривають 100% константних несправностей, є найбільш економічним рішенням завдання тестування СОК за умови реалізації схем генерації тестів з мінімальними апаратними витратами.

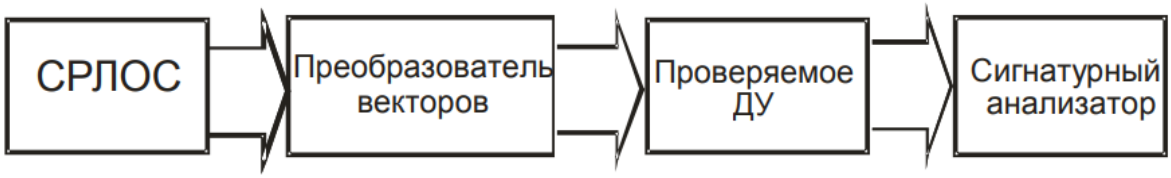


Рисунок 2.1 -Метод тестового діагностування на основі РЗЛЗЗ перетворювача тестових векторів

Одним з рішень цього завдання є використання ПЗУ, вбудованого на кристал або друковану плату, для запису і зберігання повної множини ДТ,

синтезованих системами генерації тестів і моделювання несправностей на етапі проектування СОК. Управління адресними лічильниками ПЗУ дозволяє формувати тести для послідовних схем шляхом вибірки сканованих тестових наборів, які здійснюють установку тригерів перевіряється схеми, і тестових наборів комбінаційної частини схеми, яких докладають до перевіряється схемою по закінченню сканування. Мінімізація обсягу ПЗУ для зберігання ДТ здійснюється шляхом використання методів розбиття безлічі тестів на підмножини, комбінації яких дозволяють скоротити обсяг пам'яті. При цьому ускладнюється структура схеми управління процесом діагностування, що не враховується в більшості відомих робіт. У даній роботі запропоновано новий метод побудови генераторів тестів для послідовних схем сканування, заснований на використанні генераторів гамільтонових циклів на РЗЛЗЗ і найпростіших перетворювачів тестових векторів, що формують задану множину ДТ. Основна ідея запропонованого підходу демонструється на прикладі стандартної схеми S27 (ISCAS-89), яка використовується в для побудови генератора тестів на основі ПЗУ. Передбачається, що перевірка справності схеми здійснюється в два етапи. На першому - подається безліч тестів, які формуються генератором псевдовипадкових послідовностей. На другому етапі формується безліч тестів за допомогою ПЗУ, які покривають несправності, що не виявляються тестами на першому етапі. Безліч ДТ для схеми S27 представлено в таблиці. Безліч S {011,000, 110} представляє послідовність сканованих даних, а безліч Z {0000,1101,1010,0100,0111,1001} - послідовність тестових наборів комбінаційної частини схеми.

З таблиці 2.1 випливає, що необхідно сформувати шість перевіряючих тестів, що складаються з різних комбінацій множин двійкових векторів S і Z. Нехай iz, і 1,4 - і-й розряд двійкового вектора з безлічі Z. Тоді множина трьохрозрядних векторів {z1, z2, z3} Z представляється послідовністю {000,100,110,011,101,010}, яка відповідно до алгоритму реалізації

Таблиця 2.1 - Множина тестів S27

n	s_1	s_2	s_3	z_1	z_2	z_3	z_4
1	0	1	1	0	0	0	0
2	0	1	1	1	1	0	1
3	0	0	0	1	0	1	0
4	1	1	0	0	1	0	0
5	1	1	0	0	1	1	1
6	1	1	0	1	0	0	1

гамільтонових циклів в СР формується в трьохрозрядному СР з функцією зворотного зв'язку f = $f = \overline{z_2} \oplus z_3$ (рисунок 2.2). Аналіз сканованих тестових векторів показує, що в розряді s1 рівне число 1 і 0 можна сформувати на тригері з рахунковим входом по mod2, як показано на рисунку 2.2. Далі формуються всі тестові набори в розрядах s_2, s_3 і z_4 відповідно до початкової таблицею. На наступному кроці вирішується завдання знаходження мінімальної комбінаційної схеми - формувача тестових розрядів s2, s3 і z4, входами якої можуть бути виходи тригерів (s1, z2, z3) або (z1, z2, z3). Мінімальна рішення представляється у вигляді (рисунок 2.2): .$S_1 = z_2 * z_3$, $S_1 = z_1 * z_3$, $Z_4 = z_1 \oplus z_3$

Для оцінки витрат апаратури на реалізацію вбудованих схем діагностування скористаємося оціночної методикою Synopsys Inc. Для КМОП технології виробництва інтегрованих схем (0,6 мкм, 2-х шарова металізована підкладка, 5V харчування).

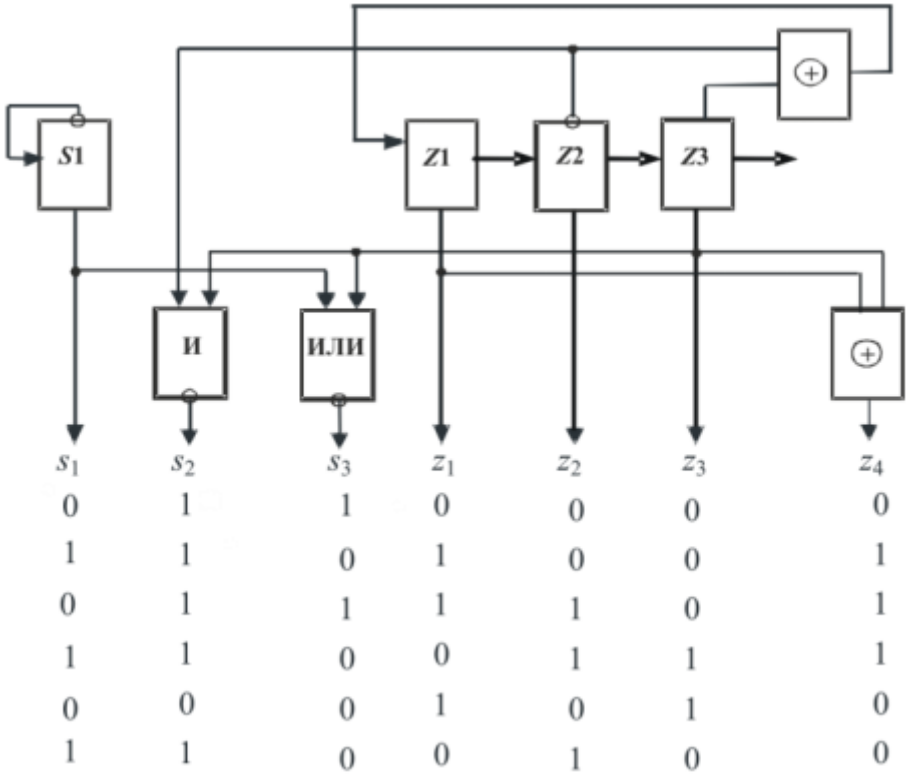


Рисунок 2.2 - Генератор тестів для еталонної схеми ISCAS-89 S27

Як одиничного вентильного еквівалента (в.е.) використовується 2-входовую I-НЕ (АБО-НЕ) елемент. Тоді витрати на реалізацію наступних елементів складають: інвертор - 0,7 В.Е .; 2-входовую I (АБО) вентиль - 1,3 В.Е .; мультиплексор 2 на 1 - 1,7 В.Е .; D-тригер - 3,6 В.Е .; XOR - 2,0 в.е.

З урахуванням того, що тригери s1, z1, z2, z3 в генераторі ДТ рисунок 2.2 є елементами вхідних регістрів СОК відповідно до стандарту проектування 1149.1, то додаткові апаратні витрати на реалізацію генератора тестів складають 6.0 в.е., а довжина тестової 8 послідовності дорівнює 6. Управління генерацією тестів зводиться до нульової початковій установці тригерів генератора і подачі 6 ти тактових імпульсів.

Для схеми S27 відповідно до методу генерації тестів з використанням ПЗУ, вбудованих на кристал, мінімальне рішення виходить у вигляді: {0,1} 1, {10,11} 2 і { (0000), (1010), (0100,0111,1001)}. Для зберігання цих даних потрібно 26 біт пам'яті. Число тестів, які послідовно формуються в 7-ми розрядному регістрі одно 12. Формування кожного тесту здійснюється шляхом зчитування даних з трьох модулів пам'яті за допомогою адресних

лічильників і пересилання даних до відповідних розряди накопичувального регістру. Очевидно, що в методі побудови генераторів тестів на ПЗУ в основні витрати на реалізацію процедури тестування необхідно включати витрати на реалізацію вбудованих засобів управління тестуванням.

Для того щоб процес тестування гарантував, що обчислювальна система не має несправних компонентів, умови тестування повинні краще розроблятися на рівні самих компонентів, ніж на рівні програмованих послідовностей. Це єдиний спосіб, у який всі стани роботи кожної логічної функції можуть бути однозначно визначені і всі логічні компоненти в кожній логічної функції виконають покладену на них завдання, таким чином виробляючи мінімальну програму, яка перевіряє і виявляє несправності.

2.3 Автоматична генерація тестових послідовностей

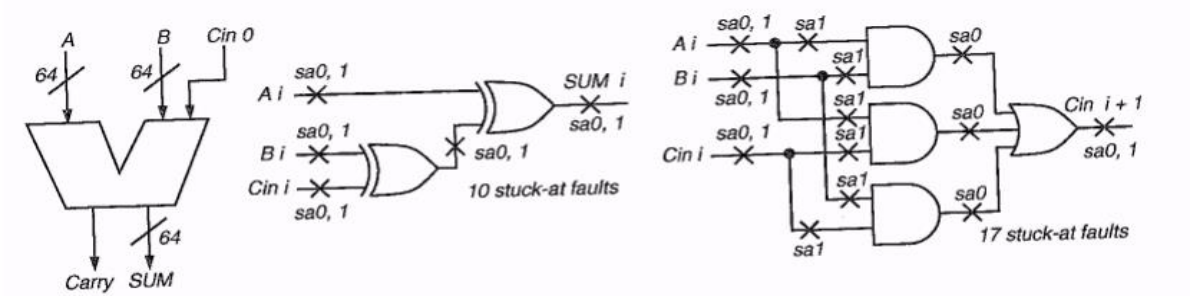
Автоматична генерація тестових послідовностей (АГТП, ATPG) - це процес створення послідовностей для тестування схеми, який точно описаний мережевий топологією логічного рівня (схематично). Ці алгоритми зазвичай працюють з програмним генератором несправностей, який створює мінімальний стислий список несправностей, таким чином, розробнику не потрібно турбуватися про генерування несправностей. У певному сенсі, АГТП-алгоритми є багатофункціональними в тому, що вони можуть генерувати тестові послідовності, вони можуть знаходити надлишкову або непотрібну схемну логіку і вони можуть довести чи одна схемна реалізація краще іншої схемної реалізації. Спочатку буде розглянуто алгоритми та подання, необхідні для АГТП. Потім - ідентифікація надмірності (ІН) - дуже важливою перевагою АГТП алгоритмів. Керованість і можливість спостереження - критерії тестованості, які використовуються в усіх головних АГТП-алгоритмах. Далі представлено декілька ключових комбінацій АГТП алгоритмів і їх поведінка на прикладах.

Спочатку про різницю між структурним і функціональним тестуванням. Поділ тестування на структурне і функціональне приписується Елдред (Richard D. Eldred). Однак, перша публікація про константної несправності логічний 0 (кн0) або 1 (кн1) для створення тестів була опублікована Гейлі (Galey), Норбі (Norby) і Ротом (Roth) в 1961. Пізніше Сешу (Seshu) і Фріман (Freeman) згадали модель константою несправності для паралельного моделювання несправностей. У 1963, Пуджа (Poage) представив теоретичний аналіз константних несправностей. Перший структурний тестовий метод використовувався, щоб перевірити Honeywell Datamatic 1000 - центральний комп'ютер другого покоління на вакуумних лампах і діодах.

Функціонально програми АГТП генерують набір тестових послідовностей, щоб повністю вивчити схемну функцію. На рисунку 2.3 зображений суматор з наскрізним переносом, і представлено дуже просте (і неефективне) логічне проектування однобітового шару суматора, якого достатньо, щоб довести дане твердження. С функціональної точки зору, суматор має 129 входів і 65 виходів. Таким чином, щоб повністю протестувати функцію, нам потрібно 2129 = 680,564,733, 841,876,926,926,749, 214,863,536,422,912 вхідних наборів, які виробляють 265 = 36,893,488, 147,419,103,232 вихідних відповідей. Якнайшвидше автоматичне тестове обладнання (АТО), в даний час, працює на частоті 1 ГГц. Це АТО має буде працювати 2.1580566142 x 1022 роки, щоб застосувати всі ці зразки до тестованої схемою (ТС), припускаючи, що тестер і схема можуть працювати на частоті 1 ГГц. Тому, ми бачимо, що вичерпне функціональне тестування неможливо, за винятком малих схем, але на сьогодні більшість схем є величезними.

Структурне тестування, з іншого боку, тільки досліджує мінімальний набір константних несправностей на кожній лінії схеми, після відкидання еквівалентних несправностей. Якщо ми використовуємо еквівалентність несправностей , то кожен бітовий шар в суматорі матиме тільки 27

несправностей (рисунок 2.3), ігноруючи еквівалентні несправності по лініях



перенесення.

Рисунок 2.3 - 64-бітний суматор: функціональний та структурний тести

Цей акумулятор не має ніяких надлишкових витрат апаратури і повний структурний список несправностей матиме не більше ніж 64 x 27 = 1,728 несправностей. Так що нам потрібно, по більшій частині, 1,728 тестових послідовностей. 1 ГГц АТО повинен застосувати ці зразки за 0.000001728 с, і так як цей набір тестових послідовностей покриває всі можливі структурні константні несправності в суматорі, це охоплює ті ж самі несправності: покриття як і в складнооброблюваному функціональному наборі тестових послідовностей, описаних вище. Часто, проектувальник схем надає обмежену підмножину функціональних тестових послідовностей для схеми, але вони зазвичай покривають тільки від 70 до 75% кількості всіх несправностей. Тестування тільки 75% модельованих помилок - це обмежене значення, так як будуть виявлені тільки найгрубіші несправності. Тому, ми бачимо важливість алгоритмів АГТП. Вектори, які вони виробляють, доповнюють функціональні тестові вектори проектувальника, що дозволяє збільшити покриття константних несправностей до рівня 98% або вище.

АГТП алгоритми вводять несправність в схему, і потім використовують різноманітні механізми, щоб активувати несправність і поширити її за схемою і спостерігати на виведенні. Вихідний сигнал відрізняється від значення справної схеми, і це дозволяє виявити

несправність. Несправність поширюється зі входу І / І-НЕ вентиля до його висновку установкою інших входів в 1, некерований значення для І / І-НЕ. Несправність поширюються зі входу АБО / АБО-НЕ вентиля до його висновку установкою інших входів в 0, некерований значення для АБО / АБО-НЕ. Несправність поширюються зі входу XOR / XNOR вентиля до його висновку установкою інших входів в 0 або 1 в залежності від ситуації.

Е-променеве тестування дозволяє спостерігати внутрішні сигнали схеми "проявляючи" зображення схеми, яке показує внутрішні точки розгалуження, встановлені в логічний 0 одним кольором і встановлені в логічну 1 іншим кольором. Це дозволяє не поширювати несправності на головні виходи (ГВих (PO)). Однак, цей метод непрактично доріг, використовується тільки для дуже спеціалізованих додатків, і в деякому сенсі перетворює складнооброблювану проблему тестування в іншу складнооброблювану проблему обробки зображень, так як деякий механізм повинен тепер переглядати зображення VLSI мікросхеми і визначати, "пофарбовані" Чи всі сигнали правильно. АГТП алгоритми надзвичайно важливі, в тому що вони поширюють несправні показання напруги з внутрішніх ланцюгів схеми до ГВих (PO), на яких АТО може досліджувати напругу і визначити, чи правильне воно.

Проектування на основі сканування для тестування мікропроцесорів. В даний час, найкращим методом для тестування щонайменше частини Intel Pentium ™ і AMD K6 ™ мікропроцесорів є комбінаційна АГТП. Вставщик скануючого ланцюжка додає мультиплексор спеціального призначення і синхронізаційних апаратуру до кожного тригеру схеми для контролю, так, щоб в режимі сканування, тригери були перетворені в великий зсувний регістр, і повне стан мікропроцесора може бути висунуто через спеціальний порт режиму тестування, званий висновок сканування. Точно так само бажаний початковий стан тригера може бути послідовно зсунути в тригер через спеціальний порт режиму тестування, що називається вхід сканування. Цей підхід перетворює складну послідовну схемну АГТП

проблему в комбінаційну схемну АГТП проблему легше піддається обробці, за рахунок:

1. використання 5 - 20% площі мікросхеми для апаратних засобів скануючих ланцюжків у великих мікросхемах.

2. уповільнення всіх тригерів через введені затримок мультиплексора скануючої ланцюжка.

3. резервування одного або більшої кількості додаткових контактів для управління скануючим ланцюжком.

4. подовження послідовності тестових наборів. Це відбувається, тому що установка машини, що має n тригерів в будь-яку бажану початкового стану вимагає n тактів скануючого ланцюжка. Застосування бажаної тестової послідовності вимагає 1 додатковий такт, наступний за n додатковими тактами читання стану тригера (коли прояв несправності зафіксовано в тригері, але не поширилося до виходу схеми.) Для множинного тестування вони можуть перетинатися.

Однак, проектування на основі сканування разом з комбінаційної АГТП - найпопулярніший метод тестування мікропроцесорів і інших VLSI мікросхем, тому що цей метод з високою ймовірністю генерує набір тестових послідовностей з близьким до 100%-ому покриттю несправностей. Крім того, час на розробку тестів передбачувано і може бути обумовлено в новому графіку впровадження продукту. Сучасна послідовна АГТП часто викликає великі затримки розробки, через проблеми алгоритмів і несумісних апаратних засобів, і може затримати впровадження продукту.

Всі програми АГТП потребують структури даних, яка описує пошуковийпростір для тестових послідовностей.

Дерева двійкового пошуку. Розглянемо бінарне дерево на рисунку 2.4 (b) для головних входів схеми (Pis) у рисунку 2.4 (a). Гоель (Goel) спочатку використовував ці дерева в комбінаційної АГТП літературі. Дерево являє всі вісім виборів для малюнків введення схеми. У найвищій точці розгалуження, якщо обрана ліва гілка, то сигнал А встановлений в 0 (гілка А), але якщо обрана права, то А в 1. У другому і третьому рівнях в дереві, наступні значення обрані для інших вводів схеми, спочатку для В і потім для С. Це покриває всі можливі вхідні послідовності. Вершини дерева позначені відповідним машинним кодом, який відповідає вхідним значенням. Все АГТП алгоритми неявно шукають це дерево, щоб знайти тестові послідовності, і в найгіршому випадку, повинні досліджувати повне дерево, щоб довести, що несправність не тестується. Ми опустимо докладне вивчення, тому що кількість листя дерева дорівнює числу основних входів і зростає по експоненті.

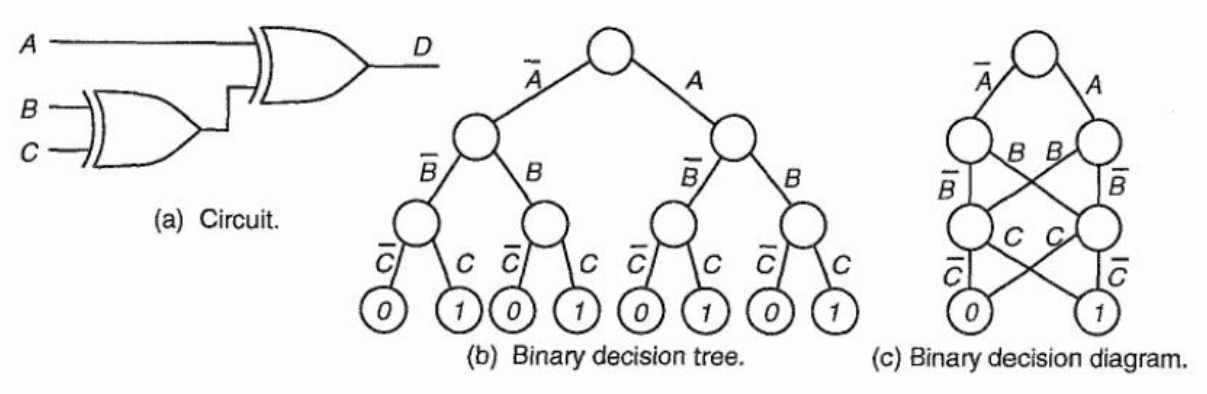


Рисунок 2.4 - Різні представлення схеми

Двійкові діаграми прийняття рішень. Будь-яка функція, що перемикається може бути повністю описана двійковій діаграмою прийняття рішень (ДДПР (BDD)), яка було винайдено Лі (Lee) в 1959. Справжнє обговорення засноване на роботі Акерс (Akers), який застосував ДДПР (BDD), щоб вирішити проблеми тестування. Рисунок 2.4 (c) показує ДДПР (BDD) для схеми на рисунку 2.4 (a). Щоб прочитати діаграму, потрібно починати в самому верхньому кореневому вузлі, і дотримуватися шляху від

цього вузла до одного з двох найнижчих вузлів, 0 або 1, який дає значення виходу схеми[8]. Добуток булевих літералів по шляху дає елементарну диз'юнктивну форму схеми або мінімальну диз'юнктивну форму схеми, в залежності від того, чи закінчується шлях в 0-му або 1-му вихідному вузлі. Наприклад, крайній лівий шлях в ДДПР (BDD) - це A B C, який виробляє 0 на виході схеми, і це збігається з функцією схеми. Шлях молодшого розряду в ДДПР (BDD) - це A B C, який виробляє 1 на виході схеми, також збігається з функцією схеми. Ми можемо перевірити, що все ДДПР (BDD) шляху сумісні з логічною функцією. ДДПР (BDD) використовувалися для АГТП, але в них є проблеми обчислювальної складності, особливо для схем множителя. Не можна не помітити суттєвих втрат обчислювального часу, в залежності від порядку в якому схемні входи описані в ДДПР (BDD).

Поняття результативності АГТП алгоритму, означає, що для того, щоб згенерувати тестову послідовність, алгоритм повинен в кінцевому рахунку бути здатний знайти повне бінарне дерево рішення, в разі необхідності, згенерувати тестову послідовність навіть для важко виявляється несправність. Якщо несправність неможливо перевірити, то після того як знаходити повного дерева ніяких тестів не знайдено. Це означає що, ця схема веде себе правильно навіть в присутності цієї несправності. Важливо для АГТП алгоритму бути закінченим, або він може не досягти необхідного покриття несправностей.

АГТП алгебра - вищий порядок подання булевої системи позначень набору з метою одночасного подання значень "справних" і "несправних" схем (або автоматів). Це дає перевагу в необхідності тільки одного проходу АГТП, щоб визначити значення сигналу для обох автоматів. Так як тестовий вектор вимагає, щоб була різниця між цими двома автоматами, з обчислювальної точки зору найшвидше - уявити ставлення обох автоматів з алгебри, замість того, щоб зберегти їх, окремими. Рот (Roth) показав, як активізація множинного шляху, вимагала протестувати деякі комбінаційні

схеми, це могло бути зроблено з його п'ятизначної алгеброю, показаної на рисунку 2.5.

Symbol	Meaning	Roth's 5-valued algebra		Muth's 9-valued algebra	
		Good machine	Failing machine	Good machine	Failing machine
D	(1/0)	1	0	1	0
$\overline{D}$	(0/1)	0	1	0	1
0	(0/0)	0	0	0	0
1	(1/1)	1	1	1	1
X	(X/X)	X	X	X	X
$G0$	(0/X)	–	–	0	X
$G1$	(1/X)	–	–	1	X
$F0$	(X/0)	–	–	X	0
$F1$	(X/1)	–	–	X	1

Рисунок 2.5 - 5-значна алгебра Рота та 9-значна алгебра Мата

Пізніше, Мат показав що, щоб тестувати кінцеві автомати, символ X повинен бути розширений, щоб покрити випадки, коли одне з справних або несправних значень автоматів може бути відомим, але інші значення автоматів - невідомі. Рисунок 2.5 показує дев'ятизначну алгебру Мата, яка також часто приносить користь комбінаційної АТГП. Для того щоб обчислити реакцію логічного вентиля, щоб ввести символи алгебри, ми розгортаємо символи в справні схемні / несправні схемні значення, дані в стовпці 2 рисунку 2.5. Ми тоді незалежно обчислюємо реакцію логічного вентиля для обох автоматів і об'єднуємо значення виходу знову в алгебру.

Різні типи АГТП алгоритми, їх ступінь інтеграції.

Вичерпний. У цьому підході, для n-вхідний схеми, ми генеруємо все 2-х входові послідовності. З причин, обговорюваних вище, це нездійсненно, якщо схема не розділена в підсхеми логічної схеми, кожна з 15 або меншою кількістю входів. Ми можемо тоді виконати вичерпне формування тестових послідовностей для кожної підсхеми. Однак, ті несправності, які вимагають

безлічі підсхем, які повинні бути активізовані синергістичним способом в процесі тестування, ця інформація не перевіряється.

Випадковий - Використовувався з алгоритмічними методами. У 1972 в Університеті Штату Іллінойс, Агравел (Agrawal) і Агравел (Agrawal) запропонував використання випадкового генерування послідовностей (ВГП, RPG) для тестування. Основна частина схеми СГП, RPG - імітатор помилки, який вибирає придатні послідовності (рисунок 2.6) При генеруванні тестів для плат паралельного комп'ютера ILLIAC IV повідомили, що покриття випадкових послідовностей буде часто варіюватися між 60 - 80% і що перемикання на D-алгоритм зумовить перевага програми в цьому сенсі. Це обмеження ВГП, RPG, яке стосується тестованості схеми, було враховано Айхельберґер (Eichelberger) і іншими для програмованих логічних матриць (ПЛМ). Стало ясно, що послідовності з однаковою ймовірністю виходів і входів, як початок ВГП, RPG на рисунку 2.6, може бути не кращим вибором. Коли ймовірності виходів і входів відрізняються від 0.5, послідовності називають зваженими випадковими послідовностями (ЗВП). Такі послідовності використовувалися Шнурманом (Schnurmann) і іншими, Ваісукавскім (Waicukauski) і Ліндблума (Lindbloom) , і деякими іншими. Метод знаходження оптимального набору ймовірностей для входів дається Вундерліха (Wunderlich).

Символічний - булева похідна. Селлерз (Sellers) та інші використовують Теорему Розгортання Шаннон, щоб характеризувати Булеві схеми. Наприклад, довільна булева функція F (X1, X2 ..., Xn) може бути розгорнута по будь-якої змінної, наприклад X2, як:

$$F(X_1, X_2, \ldots, X_n) = X_2 \cdot F(X_1, 1, \ldots, X_n) + \overline{X_2} \cdot F(X_1, 0, \ldots, X_n)$$

Предполагая, что логическая функция g = G(Xi,X2, • • •, Xn) имеет неисправность как на рисунку 2.7, мы выразим выходы как:

$$f_j = F_j(g, X_1, X_2, \ldots, X_n) \quad ; \quad 1 \le j \le m$$
$$X_i = 0 \ \ or \ \ 1 \ \ for \ \ 1 \le i \le n$$

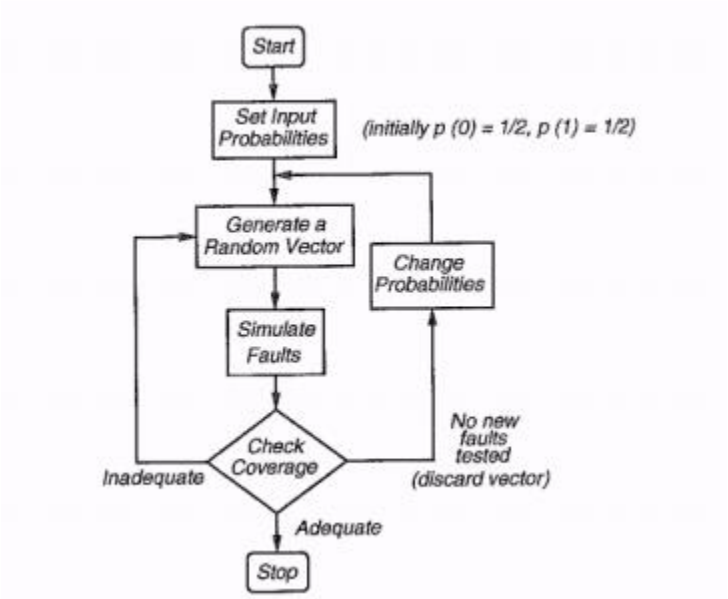


Рисунок 2.6 - Метод генерації випадкової послідовності

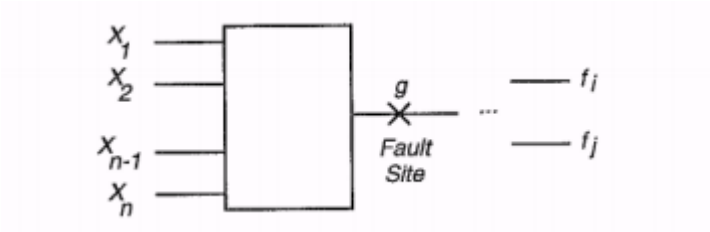


Рисунок 2.7 - Булева різниця

Важливо, що ці вихідні рівняння виражають виходи схеми в термінах безлічі первинних входів Xi і сигналу з помилкою g. Селлерз (Sellers) та інші визначили булеву похідну, або булеву часткову похідну, схеми як:

$$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \ldots, X_n) \oplus F_j(0, X_1, \ldots, X_n) \qquad (7.2)$$

Вони висловили вимоги до виявлення несправності для g КН0 на виході fj як:

$$G(X_1, X_2, \ldots, X_n) = 1$$
$$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \ldots, X_n) \oplus F_j(0, X_1, \ldots, X_n) = 1$$

Перше рівняння вище показує, що, щоб перевірити помилку константного нуля на g, логічний вентиль G повинен зробити чутливим точку помилки, встановивши її в логічну 1. Друге рівняння говорить, що, щоб виявити помилку, булева похідна деякого висновку щодо точки несправності g повинна бути 1 (тобто, вихід повинен змінити своє значення сигналу, коли сигнал точки несправності перемикається з 1 в 0). На жаль, через високу складності булева похідна - неефективний спосіб обчислювати тестові послідовності для великих схем.

Методи активізації шляхів. Активізація шляху на рівні уявлення логічного вентиля - в даний час кращий АГТП метод. Підхід складається з трьох кроків:

1. активізація помилки, при якій константна помилка активізується установкою сигналу в протилежне значення від значення помилки. Це необхідно для впевненості в поведінковому відмінності між правильною схемою і дефектної схемою. Активізація помилки також відома як активація помилки або порушення помилки.

2. поширення помилки, при якому прояв помилки поширене через один або більше шляхів до головного виходу схеми. Для деяких помилок, необхідно одночасно поширити прояв помилки по множинним шляхах, щоб протестувати її. В загальному випадку, кількість шляхів може збільшуватися по експоненті в залежності від кількості логічних вентилів у схемі. Поширення помилки також відомо як активізація шляху.

3. Обґрунтування лінії, при якому внутрішні призначення сигналу, попередньо зроблені для активізації помилки або поширення її прояви обґрунтовуються установкою головних входів схеми.

У другому і третьому етапах, ми можемо знайти конфлікт, де необхідне призначення сигналу суперечить деяким попередньо зробленим заявам. Це змушує АГТП алгоритм відстежувати в зворотному порядку або резервувати, тобто, відмовлятися від попередньо-зробленого призначення сигналу і робити альтернативне призначення.

Розглянемо приклад на рисунку 2.8. У всіх прикладах в цьому розділі, ми окреслимо головні входи і головні виходи великими літерами, і всі інші сигнальні лінії в схемі малими літерами. Зверніть увагу, що головний вхід B (основа розгалуження) розгалужується на два вентиля I, виходи яких - h і i. Розгалуження переходить від B до входів двох вентилів I зазначених f і g. Часто трапляється, що тести несправностей в B відрізняються від тестів несправностей в f, які також відрізняються від тестів несправностей на g. Саме тому ми повинні відзначати кожну окрему рядок або провід сигнальної мережі. Ми генеруємо тест для B КН0. Для активізації помилки, ми встановлюємо B в 1, і це веде до призначень сигналу f = D і g = D .

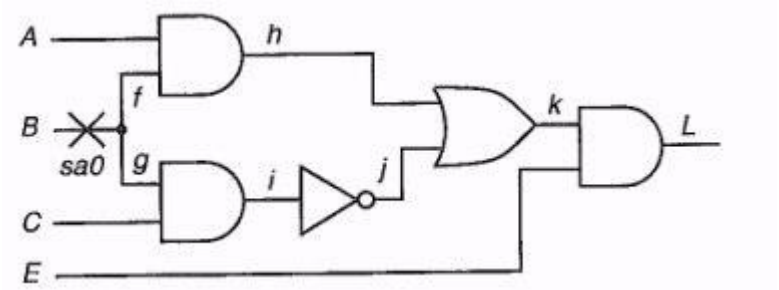


Рисунок 2.8 – Приклад комбінаційної схеми для підвищення чутливості

Поширення помилки вимагає, щоб ми вибрали один з трьох сценаріїв:

1. поширення по шляху f - h - k - L, або

2. поширення по шляху g- i - j - k - L, або

3. одночасне поширення по обом шляхах f - h - k - L і g - i - j - k - L.

Обирається шлях f - h - k - L для поширення. Це означає, що для кожного I вентиля по шляху, входи поза шляху повинні бути встановлені у некеровану значення (1), і також для кожного АБО вентиля по шляху, входи поза шляху повинні бути встановлені в 0. Це призводить до призначень сигналів A = 1 , j = 0, і E = 1. Обґрунтування лінії тепер вимагає, щоб ми обґрунтували будь-які внутрішні сигнали, яким призначалися значення. В цьому випадку, єдиний сигнал - j. Ми можемо привласнити i = 1, щоб обгрунтувати j - 0 зворотним логічним моделюванням інвертора j. Однак, I

вентиль і тоді повинен мати вихід 1, але він вже має вхід g = D. Зворотне логічне моделювання, використовуючи 5-оцінену алгебру (див. Таблицю 7.1), показує, що немає ніякого способу отримати i = 1, коли вхід вже був встановлений в D. Тому, ми відстежуємо в зворотному порядку і відмовляємося від призначення j = 0 і пробуємо альтернативне призначення j = 1. Однак, це негайно блокує поширення помилки по шляху f - h - k - L. Ми приходимо до висновку, що єдині придатні варіанти можуть бути вищезгаданими сценаріями 2 і 3.

Ми вибираємо сценарій 3. Ми змінюємо наш підхід поширення помилки, і тепер призначаємо A = 1, C = 1, і E = 1, щоб гарантувати поширення помилки. Пряме логічне моделювання результатів цих призначень i - D, h = D, j = D, і k = i, так як D АБО nD = 1. Це отримано обчисленням 1f0 АБО 0f1 = 1f1. D-кордон - це перетин, що відділяє частину схеми, позначену X від частини з позначкою D або nD, де ми включаємо тільки D або nD які мають найтісніший контакт виходів в кордоні. Нарешті, L = 1 і D-кордон зникає в АБО вентилі k, що означає, що помилка не перевіряється через множинних шляхів. Залишається тільки повернутися і пробувати поширення помилки по шляху g - i - j - k - L. Ми встановлюємо C = 1, h = 0, і E = 1, щоб поширити помилку. Пряме логічне моделювання дає i = D, j = nD, k = nD і L = nD. Залишається обгрунтувати h = 0. Це досягається зворотним логічним моделюванням для I вентиля h, з введенням f = D, встановлюючи вхід A = 0. Єдиний тест на КН0 на вході B виглядає так: ABCE = 0111, і дає вихід L = 0 в справному автоматі, і L = 1 в несправному автоматі.

Ця процедура АГТП правильна тільки для нециклічних комбінаційних схем. Ми будемо обговорювати процедури для послідовних схем в Розділі 8. Зокрема будь-яка схема з зворотними зв'язками, тригерами, або неявними тригерами-засувками, вираженими як комбінаційна логіка буде часто звертати цю процедуру в нескінченний цикл. Поширення помилки і обгрунтування лінії в АГТП складається з операцій призначень сигналів,

прямого моделювання логічного вентиля, зворотного моделювання логічного вентиля, і зворотного ходу.

Булева здійсненність і методи імплікації графів. Проблема булевої здійсненності означає задоволення булевого виразу або рівняння. N-бітний булевий вектор складається з набору n подвійних змінних, Xi, i = 1,2 ..., n. Символічно, змінна Xi або її доповнення nXi позначається як літерал. Проблема подвійного здійсненності зводиться до виявлення набору значень для rEj's, які задовольнить рівнянню типу:

$$\sum \alpha_k \beta_k = 0 \quad (\text{non} - \text{tautology}) \quad \text{or} \quad \prod(\alpha_k + \beta_k) = 1 \quad (\text{satisfiability}) \qquad (7.5)$$

де αi і βi - будь-які два літерала, і підсумовування, і множення є булеві операціями АБО і І, відповідно.

Терм або твір в рівнянні 7.5 називають булевим пропозицією або просто пропозицією. 2-SAT проблема вирішувана в поліноміальний терм. Коли пропозиції в булевому вираженні містять три літерала, проблема відома як (3-SAT) проблема з потрійною здійсненне. Рішення 3-SAT має експонентну складність часу.

Чакрадар (Chakradhar) і Ларбі (Larrabee) отримали Булевські формулювання здійсненності для АТГП проблеми. З огляду на цільову несправність, отримують функцію потужності нейронної мережі або Булевського добутку вираження сум в термінах змінних сигналу схеми такий, що будь-який тест на цільову помилку згорне функцію потужності або задовольнить булевий вираз. Мінімізація потужності показана еквівалентною булевій сумі добутків. Решта залежить від знаходження ефективних шляхів вирішення проблеми здійсненності.

Ці методи були розширені іншими, і тепер є найбільш швидкими відомими АГТП алгоритмами для великих схем. У цих методах, булева функція кожного логічного вентиля зафіксована в рівняннях, які пов'язують вхідні і вихідні сигнали вентилів.

Перевага цього підходу полягає в тому, що ми можемо записати функцію потужності для кожного логічного вентиля (або модуль булевої функції) в схемі, і потім підсумувати всі ці функції в єдину функцію потужності для всієї схеми. Якщо значення функції - 0, то всі сигнали послідовно позначені; інакше вони не відзначені.

Дійсно ефективний спосіб згорнути функцію енергії або знайти задовольняють присвоєння змінної для помилкових або істинних функцій - це граф включення. Цей граф має вузол для кожного літерала. Таким чином, булевська змінна x представлена двома вузлами, x і nx. Вузол може бути "справжній" або "хибним". Для x = 1, x вузол приймає справжній стан. Для x = 0, nx вузол стає справжнім. Вираз "якщо ... то" для двох змінних представляється як дуга від уявлення літерала умови "якщо" до подання літерала вираження "то". Граф може тоді бути перетворений в перехідній замкнутий граф так, щоб, коли вузол встановлений в істину , всі доступні вузли, також встановлені в істину. Це дозволяє дуже ефективний аналізувати значення сигналу, тому що перехідною замкнутий контур може визначити більше глобальних відносин сигналу в графі ніж інші методи пошуку гілок і меж.

Обчислювальна Складність. Ібарра (Ibarra) і Сани (Sahni) аналізували обчислювальну складність АГТП. Вони виявили, що це - NP-Complete проблема що означає, що ніяке поліноміальний вираз для функції часу обчислення не було знайдено, і проблема, як передбачається, має експонентну складність. У найгіршому випадку, з неголовним входами, є 2 різні вхідні комбінації неголовних входів, які можна пробувати для способу пошуку в глибину в довічним дереві прийняття рішень. Коли тригери присутні в схемі, є потенційно 4 різні початкові стану тригерів для АГТП, які можна розглянути. Це тому що тригери можуть бути або в 0 або в 1 стані в схемі без помилки і також в 0 або 1 стані в дефектної схемою. Таким чином, простір станів тригерів містить чотири. Нарешті, робота за прямим моделювання або зворотному моделювання всіх логічних вентилів

збільшується пропорційно n, номеру логічних вентилів. У найгіршому випадку, ця робота повинна бути зроблена для всіх потенційних комбінацій головних входів і початкових станів тригерних схем. Повний вираз для найгіршого випадку обчислювальної складності АГТП:

$$O(n * 2^{n0_{pi}} * 4^{n0\_ff})$$

Вищезазначене доказ вважає, що АГТП математично еквівалентно проблемі булевої виконуваності.

Повною історією АГТП алгоритмів був процес поліпшення евристичних алгоритмів, і процедур для (1) знаходження всіх необхідних присвоєнь сигналів для тестування якомога раніше, і (2) пошук настільки малого простору рішень, наскільки можливо. Простір рішень найгіршого випадку - 2no-pl x ^ no-ff. Для логічного моделювання обчислювальна складність становить O (n). Для комбінаційного моделювання несправності, складність становить O (n2), і для послідовного моделювання помилки, складність оцінена між O (n2) і O (n3), грунтуючись на емпіричних вимірах.

Це означає що, всякий раз, коли можливо, ми будемо використовувати моделювання несправності, щоб уникнути АГТП обчислень. Наприклад, ми використовуємо СГП, RPG і моделювання несправності, щоб отримати тести. Коли це не вдається, ми використовуємо АГТП для важко тестованих несправностей. Якщо ми знаходимо послідовність для помилки, то ми моделюємо цю послідовність замість всіх, хто лишився невиявлених помилок, в надії, що ми "випадково" перевіримо додаткові помилки. 1. величезна кількість різних несправностей, можливих у великих схемах. 2. експоненціальна складність алгоритму (тобто, для послідовної схеми тільки з 20 тригерами, послідовний АГТП може тривати дні обчислень), так як послідовний АГТП повільний на булевскому рівні уявлення, це буде навіть повільніше на аналоговому рівні уявлення. 3. АГТП для структур транзистора повинен моделювати поведінку двунаправленное, з трьома станами. Моделі несправностей і АГТП алгоритми існують, але більш складні ніж їх двійково-логічні вентилі. Хоча є тестові генератори, які

можуть працювати на рівні транзистора, переважна тестова методологія продовжує покладатися на рівень константних несправностей вентилів.

2.4 Різновиди та генерація спеціальних тестових послідовностей

Більш високої якості контролю можна досягти використовуючи ГПВТН, що спеціально орієнтовані на використання в системах контролю і дозволяють формувати тестові набори заданої ваги, тобто набори, що містять потрібну кількість одиничних символів, що розподілені між елементами набору. Використання таких алгоритмів, зокрема в системах на основі алгоритму випадкового пошуку, суттєво підвищує ефективність процедури контролю ЦВ.

Один з існуючих варіантів генератору наборів з постійною вагою представлено нижче[8].

Загальний принцип роботи такої схеми полягає в організації зсуву двійкового вектору у вихідному регістрі Reg під управлінням сигналів від звичайного генератора псевдовипадкових векторів, що формує рівноймовірні багаторозрядні вектори. Циклічний зсув вправо розрядів $Re\,g$ здійснюється в кожному такті. При цьому, якщо значення деякого розряду опорного РЗЛЗЗ дорівнює одиниці, то значення відповідного розряду Reg фіксується і цей розряд не приймає участь у виконанні зсуву.

Недоліком такого формувача є повторюваність станів протягом одного періоду.

Структура генератора. В даному генераторі управляючі сигнали для регістра Reg формуються на основі спеціальним чином відібраних булевих функцій затримки, що фактично визначають автономність такого генератора.

Нехай n – кількість розрядів регістра, тобто довжина векторів, що генеруються, k – вага кожного такого вектору, $x_1, x_2, x_3, \dots, x_n$– булеві змінні, що відображають стани відповідних розрядів регістру, а $X = (x_1, x_2, x_3, \dots, x_n)$– двійковий вектор, що генерується.

Функція f_1– булева функція, що дорівнює одиниці, якщо необхідно затримати i-й розряд регістра Reg і нулю якщо навпаки. варто відмітити, що в загальному випадку функція затримки може залежати не тільки від значень розрядів генератора.

Таким чином, значення i-го роз $(t+1)$ такті визначається виразом :

$$x_i|t+1| = x_l|t|,$$

де $l = \max\limits_{1 \le i < j}(j \mid f_j = 0)$, якщо $\prod_{j=1}^{i-1} f_j = 0$ і $l = \max\limits_{i \le j \le n}(j \mid f_j = 0)$, якщо

$\prod_{j=1}^{i-1} f_j \neq 0$, а f_j – значення функції затримки для j-го розряду генератора в даний такт t. На рисунку 2.8 представлена схема такого генератора в загальному вигляді.

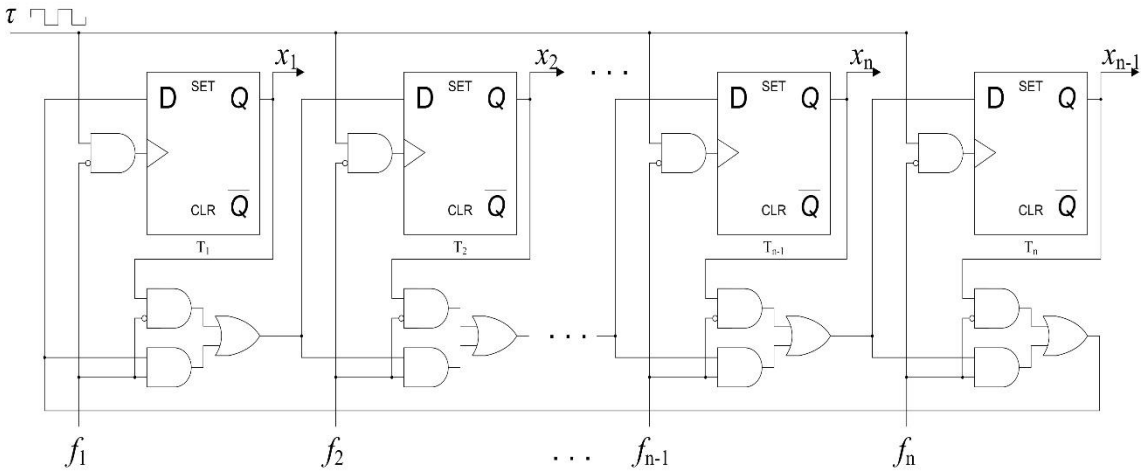


Рисунок 2.8- Загальна схема генератора

Основною задачею при створені даного автономного генератора псевдовипадкових векторів постійної ваги, яких не повторює свої стани протягом одного періоду, є визначення функції затримки.

Далі буде розглядатися окремий випадок генератора, у якого фактично визначена функція затримки f тільки для першого розряду, а всі інші за замовчування дорівнюють нулю.

2.3.2 Властивості генератора

Можна впевнитись, що даний генератор може сформувати всі двійкові вектори певної довжиниn та ваги k.

Нехай B_n –множина всіх двійкових векторів довжиною n, B_n^k – множина всіх двійкових векторів довжиною k, $B_n^k \subset B_n$.

Тоді через Sh $(Shif\ t)$ буде позначено операцію циклічного зсуву,

$$Sh: B_n^k \rightarrow B_n^k,$$
$$Sh(X) = Sh(x_1, x_2, \dots, x_{n-1}, x_n) = (x_n, x_1, \dots, x_{n-2}, x_{n-1}),$$
$$Sh^i(X) = Sh(X)(Sh^{i-1}(X)),$$
$$Sh^1(X) = Sh(X).$$

Також, для спрощення подальших розрахунків визначено

$$Sh^{-1}(X) = (x_2, x_3, \dots, x_n, x_1) = X'$$

тобто $Sh(X') = X$. А операцію циклічного зсуву зсуву з затримкою визначено як $D\ s$ $(delay\ s\ hift)$,

$$Ds(X) = Ds(x_1, x_2, \dots, x_{n-1}, x_n) = (x_1, x_n, x_2, \dots, x_{n-1}),$$
$$Ds^i(X) = Ds(X)(Ds^{i-1}(X)),$$
$$Ds^1(X) = DS(X),$$
$$Ds^{-1}(X) = (x_1, x_3, \dots, x_n, x_2).$$

Легко побачити, що вирішення поставленої вище задачі можна звести до доведення наступного твердження.

Твердження 1. З довільного вектору $X_i \in B_n^k$ можна отримати будь-який інший вектор $X_j \in B_n^k$ шляхом виконання ряду операцій Ds та/або Sh.

Для доведення цього спочатку варто довести наступне положення. Нехай є вектори X_0, $X_T \in B_n^k$, та $\forall\ l\ (l \neq \alpha, l \neq \beta)$:

$$x_l^0 = x_l^T, \qquad x_\alpha^0 = x_\beta^T, \qquad x_\beta^0 = x_\alpha^T,$$

де x_l^0– значення l-го элемента вектору X_0, x_l^T– значение l-го элемента вектору X_T, $\alpha, \beta \in [1 \dots n]$. Шляхом виконання ряду операції Ds та $S\,h$ з вектору

X_0 можна отримати вектор X_T. Це продемонстровано далі:

$$\begin{aligned}
(x_1, .., x_{\alpha-1}, x_\alpha, x_{\alpha+1}, \dots, x_{\beta-1}, x_\beta, x_{\beta+1}, \dots, x_n) &= X_0,\\
(x_\beta, x_{\beta+1}, .., x_n, x_1, \dots, x_{\alpha-1}, x_\alpha, x_{\alpha+1}, \dots, x_{\beta-1},) &= Sh^{n-\beta}(X_0) = X_1,\\
(x_\beta, x_\alpha, x_{\alpha+1}, \dots, x_{\beta-1}, x_{\beta+1}, .., x_n, x_1, \dots, x_{\alpha-1}) &= Ds^{\beta-\alpha+1}(X_1) = X_2,\\
(x_\alpha, x_{\alpha+1}, \dots, x_{\beta-1}, x_{\beta+1}, .., x_n, x_1, \dots, x_{\alpha-1}, x_\beta) &= Sh^{-1}(X_2) = X_3,\\
(x_\alpha, x_{\beta+1}, .., x_n, x_1, \dots, x_{\alpha-1}, x_\beta, x_{\alpha+1}, \dots, x_{\beta-1}) &= Ds^{-(\beta-\alpha+1)}(X_3,) = X_4,\\
(x_1, \dots, x_{\alpha-1}, x_\beta, x_{\alpha+1}, \dots, x_{\beta-1}, x_\alpha, x_{\beta+1}, .., x_n) &= Sh^{-(n-\beta)}(X_4) = X_5.
\end{aligned}$$

Отриманий вектор X_5 і є шуканим вектором X_T. Таким чином, можна стверджувати, що з будь-якого вектору можна отримати інший вектор, що відрізняється від початкового перестановкою значень тільки двох розрядів, шляхом виконання послідовності операцій Ds та/або Sh.

Далі, нехай є вектори $X_i, X_j \in B_n^k$. Оскільки обидва ці вектори мають однакову вагу, то шляхом послідовної попарної перестановки розрядів вектору X_i ,значення яких не співпадають зі значеннями відповідних розрядів в X_j, у відповідності з доведеним положенням завжди можна перетворити вектор X_i в X_j. Твердження 1 доведено.

Твердження 1 підтверджує, що на основі даного генератора може бути сформована послідовність векторів, яка містить всі вектори з B_n^k. Проте в такій послідовності не виключені повторення .

2.3.3 Функція затримки

Нехай F визначено як множину функцій затримки$f(X)$, таких, що для $\forall\, f(X) \in F$ період генератора максимальний і дорівнює $\left|B_n^k\right| = \mathrm{C}_n^k$ тактів. Ця вимога еквівалентна тому, що

$$X_i \neq X_i \ (i \neq j) \text{ для будь яких } i\ , j\ \in 1 \dots \mathrm{C}_n^k$$

де i, j – номера векторів в послідовності, що генерується . Далі під позначенням f буде розумітися $f(X)$ тобто функцію затримки, яка залежить тільки від розрядів генератора.

Далі, шляхом доведення ряду тверджень можна побачити, що якщо функція затримки $f \in F$ існує, то вона не залежить від значень першого і останнього розрядів генератора.

Перш за все варто відмітити, що функція затримки є частково визначеною, оскільки не визначена на множині векторів $B_n \backslash B_n^k$. Крім цього можна побачити, що значення функції $f \in F$ не визначено на деяких векторах множини B_n^k.

Потрібно пам'ятати, що при виконані операції $Ds(X)f(X) = 1$, і при $Sh(X)\, f(X) = 0$. Згідно з відомим твердженням (ст мет) якщо $x_1 \oplus x_n = 0$, то $Ds(X) = Sh(X)$, і результат не залежить від виконуваної операції, а значить і від значення функції на векторі X. Це доводить:

Твердження 2. Для $f \in F$, значення $f(x_1, x_2, \ldots, x_{n-1}, x_n)$ можна вважати невизначеним, якщо $x_1 \oplus x_n = 0$.

Для простоти, вектори $X = (x_1, x_2, \ldots, x_{n-1}, x_n)$, в которых $x_1 \oplus x_n = 1$, будуть мати назву перехідні. Нехай $X_1 \in B_n^k$ – перехідний вектор, тоді нехай $X_2 = Sh(X_1), X_4 = Ds(X_1), X_3 = Sh^{-1}(X_4)$

Далі можна розглянути дану четвірку попарно різних векторів детальніше (рисунок 2.11).В загальному випадку з вектору X_i можна перейти в вектор X_j, якщо $X_j = Ds(X_i)$ або $X_j = Sh(X_i)$.

Доведемо, что $Ds(\mathbf{X}_3) = \mathbf{X}_2$. Якщо $\mathbf{X}_1 = (x_1, x_2, \ldots, x_{n-1}, x_n)$, тоді $Sh(\mathbf{X}_1) = \mathbf{X}_2 = (x_n, x_1, x_2, \ldots, x_{n-1})$. $Ds(\mathbf{X}_1) = \mathbf{X}_4 = (x_1, x_n, x_2, \ldots, x_{n-1})$, $\mathbf{X}_3 = Sh^{-1}(\mathbf{X}_4) = (x_n, x_2, \ldots, x_{n-1}, x_1)$.

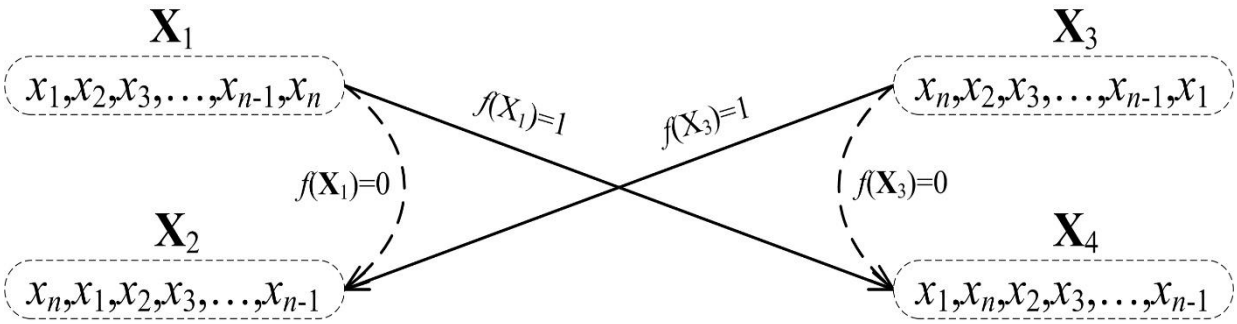


Рисунок 2.11 Можливі переходи між векторами і групами

Доведення, що $Ds(X_3) = X_2$. Якщо $X_1 = (x_1, x_2, \dots, x_{n-1}, x_n)$, тоді

$$Sh(X_1) = X_2 = (x_n, x_1, x_2, \dots, x_{n-1}),$$
$$Ds(X_1) = X_4 = (x_1, x_n, x_2, \dots, x_{n-1}),$$
$$X_3 = Sh^{-1}(X_4) = (x_n, x_2, \dots, x_{n-1}, x_1),$$
$$Ds(X_3) = (x_n, x_1, x_2, \dots, x_{n-1}).$$

і, дійсно, отриманий вектор дорівнює X_2.

В вектор X_2 можна перейти тільки з X_1 або з X_3. Нехай $f'(X_1) = 0$, $f'(X_3) = 1$, тоді $Sh(X_1)=Ds(X_3)=X_2$, отже, в послідовності генерованих під впливом функції f' векторів, вектор X_2 з'явиться двічі, отже, $f' \notin F$. Аналогічно, якщо $f'(X_1) = 1, f'(X_3) = 0$, то $f' \notin F$.

Таким чином, якщо функція $f \in F$ то $f(X_1) = f(X_3)$. Варто відмітити, що вектори X_1 та X_3 відрізняються тільки значення 1-го та n-го розрядів. Це підтверджує справедливість:

Твердження 3. Для будь-якої $f \in F$ виконується : $f(x_1, x_2, \dots, x_{n-1}, x_n) = f(x_n, x_2, \dots, x_{n-1}, x_1)$, якщо $x_1 \oplus x_n = 1$.

Основуючись на твердженнях 3 і 4, можна зробити висновок , що функція затримки $f \in F$ дійсно не залежить від значень розрядів x_1 і x_n векторів $(x_1, x_2, \dots, x_{n-1}, x_n) \in B_n^k$ (тобто фактично не залежить від значень першого і останнього розряду генератора).

Нехай $Y=(x_2, x_3,\ldots, x_{n-1})$. Далі разом з позначенням $f(X)$ буде використовуватись $f(Y)$, причому

$$f(Y) = f\big((0) + Y + (1)\big) = f((1) + Y + (0)),$$

при тому. що знаком «+» позначена операція конкатинації над векторами.

На основі вищесказаного можна сказати, що значення функції f фактично не визначено на векторах $Y \in B_{n-2} / B_{n-2}^{k-1}$, що певним чином спрощує форму і реалізацію функції затримки. На основі цієї властивості можна спростити досконалу диз'юнктивну нормальну форму функції, і отримати ДНФ з термів(які відповідають векторам Y при $f(Y) = 1$) наступного вигляду[9]:

$$K = \prod_{x_i=1, x_i \in Y} x_i.$$

2.5 Висновки

У даному розглянуто використання випадкових і псевдовипадкових послідовностей в якості тестових послідовностей. На практиці розповсюджені апаратно генеровані псевдовипадкові тестові набори великої довжини. Вони використовуються в системах контролю, що основуються на методах випадкового пошуку, сигнатурного аналізу, реалізованих на ймовірнісних принципах, а також при організації самотестування великих і надвеликих інтегральних схем. При цьому в якості псевдовипадкових тестових наборів в більшості випадків використовуються М-послідовності. Однак такий підхід обмежує кількість цифрових вузлів, що можна контролювати. Причиною цього є детермінованість структури даних псевдовипадкових тестових наборів, яка виражається в залежності між її символами.

АГТП-алгоритми є багатофункціональними в тому, що вони можуть генерувати тестові послідовності, вони можуть знаходити надлишкову або непотрібну схемну логіку і вони можуть довести чи одна схемна реалізація іншої схемної реалізації.

Значного розповсюдження набули псевдовипадкові послідовності постійної ваги, адже в більшості випадків не надається можливим провести експеримент з моделлю на всій множині векторів стану системи в зв'язку з тим, що системи можуть мати велике число складових модулів.

ОПИС МЕТОДУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Основні положення нового методу

Одним з головних типів послідовностей двійкових векторів, що використовуються в тестах з моделями поведінки цифрових систем є послідовності векторів сталої ваги. Даний метод дозволяє створювати генератори, що генерують саме такі послідовності. Проте даний метод розрахований для створення генераторів спеціально для використання у тестах з моделями неоднорідних систем. Оскільки неоднорідність системи полягає в тому, що її компоненти мають різну ймовірність вийти з ладу, то для оптимізації тестування доцільно використовувати набори, у яких одиничні значення, що відповідають виходу з ладу компонента, з'являлися на позиціях елементів, що мають більшу ймовірність вийти з ладу.

За основу береться вищезгадана схема генератора ПВ-векторів постійної ваги, але її схему управління зсувом модифікується так, щоб у зсуві приймали участь тільки ті розряди регістру зсуву, для яких значення певної функції збудження $f_i = 1,\ i = \overline{1,n},$ де n– розрядність регістра. Така модифікована схема буде використовуватися як вихідний компонент генератора. Така модифікація є першою особливістю нового методу. Головна ідея полягає в тому, що якщо подавати на вихідну схему такі функції f_i, які тим частіше приймають значення 1, чим менше значення індексу i_i. Це призведе до того, що значення відповідних розрядів регістру зсуву будуть частіше змінюватися і це дозволить сконцентрувати в них одиничні значення набору вихідного вектору.

Тому першою стоїть задача отримати такі вектори функцій збудження $F = \{f_1, f_2, f_3, \ldots, f_i\}$, у яких $\nu f_i{=}1 \geq \nu f_{i+1}{=}1$, де $\nu f_i{=}1$ - частота, з якою функція f_i приймає одиничне значення. Одним з можливих рішень є використання такого ж регістра зсуву з лінійним зворотним зв'язком з модифікованою схемою управління зсувом, для якого вектори збудження буде формуват генератор псевдовипадкових чисел розрядністю m з рівномірним розподілом ймовірностей появи наборів. В такій схемі

ймовірність появи одиничного значення на будь-якій позиції РЗ буде залежати від значення ваги його набору. Якщо з'єднати n таких схем і взяти з кожного регістру зсуву значення одного розряду, наприклад першого, можна отримати вектор функцій збудження F для вихідної схеми генератора. Ваги наборів у таких регістрах будуть відповідати ймовірностям виходу з ладу елементів певної неоднорідної системи.

Якщо впорядкувати всі ці значення ймовірностей відмов у $\{p_1, p_2, p_3, \dots, p_i\}, p_i \geq p_{i+1}, i = 1, n-1$, де p_i- ймовірність виходу з ладу і-того елементу, n – кількість елементів системи. Проте абсолютне значення вхідних ймовірностей може бути надзвичайно малим. Тому їх потрібно певним чином нормалізувати.

Для цього нехай $p_i^{'}$ - значення нормалізованої ймовірності *i*-го розряду, тоді:

$$p_i^{'} = \left(\frac{p_i}{p_1}\right) * b\,, \qquad i = \underline{1\,, n}$$

де b – база нормалізації.

Проте, ця єдина модифікація не сильно вплине на вигляд результуючого розподілу, адже ймовірність знаходження одиничного розряду на певній позиції можна описати формулою:

$$p_i = p_s \frac{k}{n} + p_{ns} \frac{k}{k} = (p_s + p_{ns}) \frac{k}{n} = \frac{k}{n}\,,$$

де p_s- ймовірність того, що розряд приймає участь у зсуві, p_{ns}- ймовірність того, що розряд не приймає участь у зсуві, k - вага вектора, n - довжина вектора. Тобто, ця формула показує, що ймовірність знаходження одиничного значення у розряді дорівнює сумі ймовірного того, що розряд прийме участь у зсуві, у якому на його місце стане одиниця і і не важливо яке значення розряд мав до того і ймовірності того, що у розряді вже знаходилась одиниця і він не приймає участь у зсуві. Сама по собі ймовірність зсуву не впливає на ймовірність знаходження одиничного значення, адже при потраплянні одиниці або нуля вихідного регістру в розряди, які мають невелику

ймовірність прийняти участь у зсуві, вони там затримуються. І навпаки, розряд, що часто зсувається буде часто змінювати своє значення і в обох випадках ймовірність отримати одиничне значення у розряді буде приблизно однаковою.

Тому другою особливістю методу є додавання схеми, що буде вибирати одиничне значення з розрядів, яким відповідає менша ймовірність прийняти участь у зсуві, та нульове значення з розрядів, що мають більшу ймовірність прийняти участь у зсуві, а потім взаємозамінити їх. Відбуватися така перестановка буде в залежності від відношення між найбільшим значенням вектору вхідних ймовірностей і найменшим. Чим більше різниця між ними, тим частіше буде відбуватися взаємозаміна, але не частіше, ніж кожний другий такт зсуву вихідної послідовності, адже інакше одиниця в останньому розряді вихідного РЗ взагалі не буде з’являтися.

3.2 Опис програмного продукту

На основі розробленого методу, викладеного вище, було розроблено програмну модель генератора псевдовипадкових двійкових векторів постійної ваги для використання у тестах з моделями поведінки неоднорідних цифрових систем у потоці відмов. Основною задачею моделі є формування псевдовипадкових послідовностей заданої ваги, але зі змінною ймовірністю появи наборів.

Програма моделі складається з модулів.

Опис модулів програми:

1. Управляючий модуль: ядро програми, що відповідає за керування іншими модулями.
2. Модуль регістру зсуву: описує структуру і функціоналу регістру зсуву.
3. Модуль генерації вхідних ймовірностей: формування тестових ймовірностей, їх нормалізація.

Модуль генератора: опис ініціалізації вихідного регістра, формування нового вектору функцій збудження, основних функцій генератора.

4. Модуль тестування генератора: визначення початкових значень та тестовий цикл генератора, формування файлів результату.

5. Модуль графічного представлення результатів: створення графіків для відображення вхідного розподілу ймовірностей та вихідного розподілу одиничних значень.

При запуску програмного продукту створюється вікно, у якому знаходяться поля параметрів для ініціалізації тесту, та копка ‘Start’, яка при натисканні запускає тест з відповідними параметрами. Вікно програми зображено на рисунок 3.1.

За запуск генерації відповідає функція test.

Детальніше про поля параметрів початкового вікна:

поле ‘Probability vector’ відображає вектор вхідних ймовірностей відмов;

- поле ‘Vector weight’ відповідає за вагу векторів, що будуть згенеровані;

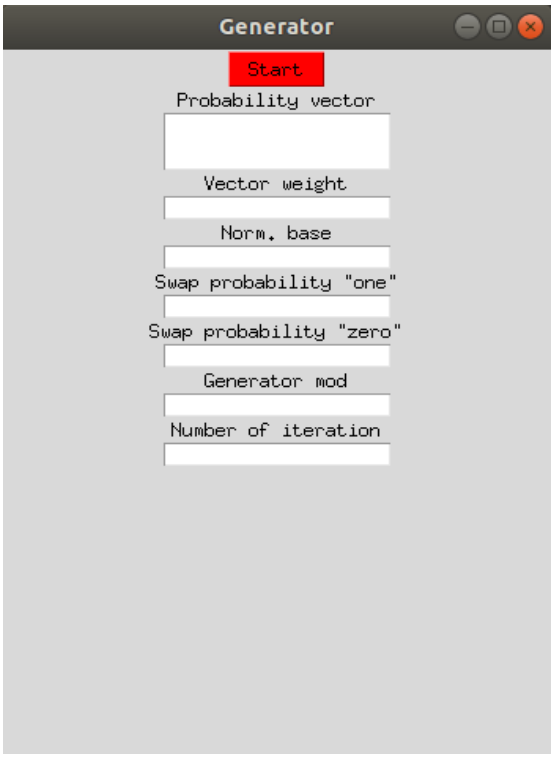


Рисунок 3.1 - Початкове вікно програми

- поле ‘Swap probability “one”’ відповідає за вибір одиничного значення для взаємозаміни;

- поле ‘Swap probability “zero”’ відповідає за вибір нульового значення для взаємозаміни;

- поле ‘Generator mod’ відповідає за режим формування векторів функцій збудження;

- поле ‘Number of iterations’ відповідає за кількість повторень циклу, тобто за кількість згенерованих векторів;

Якщо в поля були введені некоректні дані, користувач отримає повідомлення про помилку (рисунок 3.2).

Через те, що для перевірки роботи генератора потрібний вхідний вектор ймовірностей відмов, ці вектори можна можна генерувати програмно. За це відповідає функція test_prob_array():

```
part_list = [1, 0.5, 0.2]

res_list = []

print('-init_test_probs  list (10, base*10**(-3), 50)\n')

for base in part_list:

   res_list +=  init_test_probs_test(10, base*10**(-3), 50)

print(sorted(res_list, reverse = True))

print('\nrand_list\n')

rand_list = []

for i in range(20):
   rand_list.append(round(uniform(0,10**(-3)), 10))


print(sorted(rand_list, reverse = True))
```

Вектор ймовірностей буде складатися з найбільшої ймовірності base та ймовірностей, що будуть зменшуватись і відрізнятись одна від одної на

значення не менше, ніж (probs_array[i-1])/step)/2 та не більше, ніж 1.5*((probs_array[i-1])/step), де probs_array[i-1] – значення попередньої ймовірності. Для розрахунку використано функцію uniform модулю random для Python, що з рівною ймовірністю повертає випадкове значення числа з плаваючою крапкою з заданого діапазону. Завдяки цьому формується впорядкований вектор тестових ймовірностей.

Для використання вхідні ймовірності повинні бути нормалізовані. Для цього використовується функція probability_norm.

```
def probability_norm(prob_array, inner_reg_len, coef, ):

    l = len(prob_array)
    prob_weights = []
    n = inner_reg_len
    prob_weights.append(n-1)
    for i in range(1, l, 1):
        tmp = round((n-1)*(prob_array[i]/prob_array[0]))
        prob_weights.append(tmp)
        if prob_weights[i] == 0 :
            prob_weights[i] = 1
    prob_weights[0] = round(coef*n)
    return prob_weights
```

Проте значення першої і, відповідно, максимальної нормалізованої ймовірності множиться на base. Це зроблено для можливості додаткових налаштувань розподілення вихідних послідовностей.

Для ініціалізації початкового значення вихідного вектора використовується функція reg_init(inner_reg_len, mod). В якості параметрів вона приймає довжину вихідного вектора та значення mod, ща вказує в якомі режимі буде працювати функція і в якому вигляді буде сформовано значення вихідного вектора.

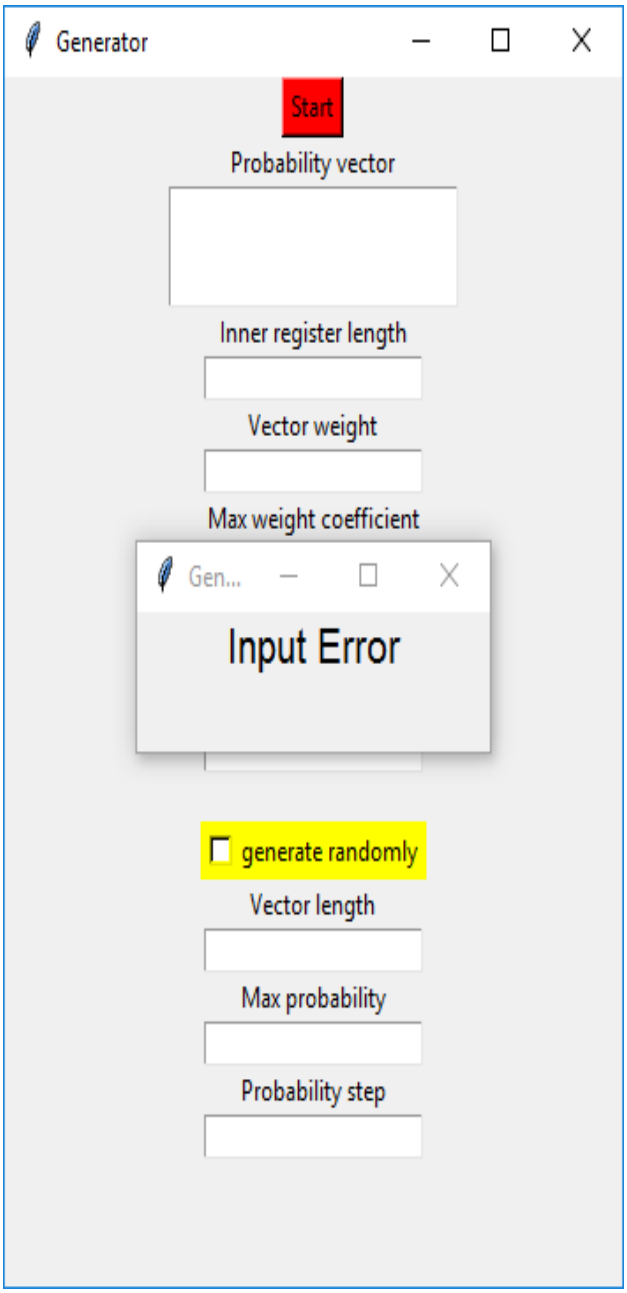


Рисунок. 3.2 - Повідомлення про помилку вводу вхідних даних

```
def reg_init(inner_reg_len, mod):

    tmp_inner_reg = []
    in_prob_weight = prob_weight
    ind = 0

    if mod == 1:

if mod == 'random':
        for j in range (inner_reg_len):
            tmp_inner_reg.append(0)

        while in_prob_weight  > 0:
            ind = randrange (0, inner_reg_len, 1)
            if tmp_inner_reg [ind] != 1:
                tmp_inner_reg[ind] = 1
                in_prob_weight -= 1
    elif mod == 'notrandom':
        for i in range(inner_reg_len):
            if in_prob_weight - i > 0:
                tmp_inner_reg.append(1)
            else:
                tmp_inner_reg.append(0)
    else:
            print('generator, reg_init mod error')
    return tmp_inner_reg
```

Одним з варіантів запису значення вхідного вектору є запис спочатку всіх одиниць потім всіх нулів. Другим варіантом є циклічне заповнення одиницями поки значення ваги більше нуля і потім вставлення нулів на всі інші позиції.

Для розрахунку використана функція randrange модулю random для Python. Вона з рівною ймовірністю повертає цілочисельне значення з певного діапазону. Функція reg_init також використовується для формування значення вихідної послідовності.

Самі регістри зсуву є об’єктами класу Shift_Register. У такого об’єкта є поле даних що містить значення вектора, можливість встановити значення

певного розряду, отримати значення певного розряду та отримати значення всього регістру. Також для них визначені операції керованого зсуву shift та взаємозаміни correct_swap:

class Shift_Register:

```
def reg_init(inner_reg_len, mod):

    tmp_inner_reg = []
    in_prob_weight = prob_weight
    ind = 0

    if mod == 1:

            class Shift_Register:

                def __init__(self, start_pattern = []):
                    self.reg_data = start_pattern

                def shift(self, pr_sequence):
                    inner_reg = {}

                    for i in range(len(pr_sequence)):
                        if pr_sequence[i] == 1:
                            inner_reg[i] = self.reg_data[i]

                    pos_list = list(inner_reg.keys())
                    if len(pos_list) > 1:
                        tmp = inner_reg[pos_list[len(pos_list)-1]]

                        for i in range(len(pos_list)-2, -1, -1):
                                inner_reg[pos_list[i+1]] = inner_reg[pos_list[i]]
                        inner_reg[pos_list[0]] = tmp
```

```
j = 0
for i in range(len(self.reg_data)):
    if i == pos_list[j]:
        self.reg_data[i] = inner_reg[pos_list[j]]
        j += 1
        if j == (len(pos_list)):
            break



    def correct_swap(self, pr):


        for i in range(len(self.reg_data)-1, -1, -1):
            if (self.reg_data[i] == 1) and (randrange(0, 100, 1) > pr):
                break
        for j in range(len(self.reg_data)):
            if self.reg_data[j] == 0 and j < i:
                self.reg_data[j] = 1
                self.reg_data[i] = 0
                break
```

Функція shift викнує зсув значень регістру на основі певної послідовності. У ролі генератора псевдовипадкових двійкових послідовностей з рівномірним розподілом слугує функція get_prand_seq:

```
def get_prand_seq  (reg_len):

    pr_sequence = []
    for j in range(reg_len):

        pr_sequence.append(randrange(0,2,1))
    return pr_sequence
```

Вона використовує вищезгадану функцію randrange. Такі послідовності використовуються в якості векторів збудження для зсуву значень внутрішніх регістрів. Для зсуву ж вихідного регістру у якості вектору збудження використовують значення вектору внутрішніх регістрів і за це відповідає функція out_reg_shift:

```
def (out_ out_reg_shift reg, inner_regs_array, inner_reg_len, correct_swap_key, pos_pr):

    inner_reg_len
    inciting_array = []
    pr_sequence =  []

    for i in range(len (inner_regs_array)):
        pr_sequence = get_prand_seq(inner_reg_len)
        inner_regs_array[i].shift(pr_sequence)
        inciting_array.append(inner_regs_array[i].get_elem(0))

out_reg.shift(inciting_array)
    if correct_swap_key == True:
        out_reg.correct_swap(pos_pr)
```

Функція out_reg_shift після зсуву значень вихідного вектору може виконати коригуючу перестановку correct_swap, що показана вище. Correct_swap виконує перестановку першого нуля зліва та одиниці справа. Яка саме одиниця буде переставлена залежить від значення ‘Swap change probability’. Починаючи з лівого кінця береться одиниця і переставляється з ймовірністю (‘Swap probability’)%. Якщо одиниця не переставилась, береться наступна ліва і т. д. Переставитися можуть одиниці лише справа

від вибраного нуля. Це також зроблено для додаткової можливості налаштовувати розподілу одиниць вихідних наборів.

Після завершення формування вектору внутрішніх регістрів і вихідного регістру обчислюється значення cor_sw_ind, що показує чи необхідно проводити коригуючу перестановку вихідного вектору на даному такті. cor_sw_ind залежить від значення мінімальної з ваг наборів внутрішніх регістрів, але не може бути меншим за 2. Це все виконується в функції test:

```
def test(inner_reg_len, out_reg_len, weight, max_prob, prob_step, start_pattern_init_mod,
         max_prob_weight_coef, inner_regs_init_mod, sw_pos_pr, iter_num):
    out_reg_file= open('out_reg_log.txt', 'w')
    res_file = open('res_file.txt', 'w')

    start_pattern =[]
    inner_regs_array =[]
    res_array =[]
    prob_weights =[]
    probs_array =[]
    probs_array = nm.init_test_probs(out_reg_len, max_prob, prob_step)
    start_pattern = gt.reg_init(weight, out_reg_len, start_pattern_init_mod)
    out_reg = gt.out_reg_init(start_pattern)
        prob_weights = nm.probability_norm(probs_array, inner_reg_len, max_prob_weight_coef)
        inner_regs_array = gt.inner_regs_init (prob_weights, inner_reg_len, inner_regs_init_mod)

    if prob_weights[len(prob_weights)-1] > 3:
        cor_sw_ind = (prob_weights[len(prob_weights)-1]) // 2
```

```
else:
    cor_sw_ind = 2
```

Після ініціалізація всіх необхідних даних на основі значень, що були введені в поля вводу в стартовому вікні, як було описано вище, запускається цикл розрахунку результуючих значень вихідної послідовності.

```
for i in range(len(out_reg.reg_data)):
    res_array.append(out_reg.get_elem(i))
```

Після розрахунку всіх значень, згенеровані вектори двійкових наборів, вектор ймовірностей відмов елементів, вектор ваг внутрішніх регістрів та вектор, що містить кількості появ одиничних значень на кожній позиції вихідного набору, записуються у відповідні файли. Дані значення записуються в файли, а не виводяться на екран через те, що їх може бути дуже багато.

```
out_reg_file.write(str (out_reg.reg_data))
out_reg_file.write('\n')
for i in range(iter_num):
    if i % cor_sw_ind  == 0:
            gt.out_reg_shift(out_reg, inner_regs_array, inner_reg_len, True, sw_pos_pr)
    else:
            gt.out_reg_shift(out_reg, inner_regs_array, inner_reg_len, False, sw_pos_pr)
    out_reg_file.write(str(out_reg.reg_data))
    out_reg_file.write('\n')
    for i in range(len (out_reg.reg_data)):
        if out_reg.get_elem(i) == 1:
            res_array[i] += 1
res_file.write ('res array      prob weights      prob array  \n')
for i in range(len(out_reg.reg_data)):
```

```
                res_file.write('{0:10d}    {1:10d}    {2:11s}'.format(res_array[i],
prob_weights[i], ' ') + eformat(probs_array [i], 5, 3)+ '\n')
            #res_file.write(str(probs_array[i]) '\n')
            #res_file.write('prob weights\n')
            #res_file.write(str(prob_weights [i])+ '\n')
        res_file.close
        out_reg_file.close
        pl.plot_results(probs_array, res_array)
```

Після запису обчислення і запису результатів виводяться 2 графіка, які показують, наскільки розподілення значень вихідного вектору відрізняється від розподілу значень тестових ймовірностей. За формування графіків відповідає функція plot_results:

```
    def plot_results(probs_array, res_array):
        w = 200
        h = 60

        sw = err_w.winfo_screenwidth()
        sh = err_w.winfo_screenheight()
        x = (sw - w)/2
        y = (sh - h)/2
        err_w.geometry('%dx%d+%d+%d' % (w, h, x, y))

    def try_get(self, in_arg):
      try:
         return eval(in_arg)
      except:
         self.err_win


      x = np.arange(0, len(probs_array), 1)
    plt.figure(1)
    plt.subplot(211)
    plt.xlabel('Pocessors')
    plt.ylabel('Faults probabilities')
```

```
plt.plot(x, probs_array)
plt.show()
```

3.3 Результати роботи моделі та їх аналіз

Основними результатами роботи моделі є файл зі згенерованими векторами out_reg_log, файл зі статистикою появи одиничних значень на відповідних позиціях, вектором вхідних ймовірностей та вектором нормалізованих ймовірностей res_log та графіки, що відображають вхідні ймовірності та розподіл частоти появи одиничних значень.

Приклади результатів представлено нижче на рисунку 3.2, рисунку 3.3 та рисунку 3.4 відповідно.

```
[0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Рисунок 3.2 - Приклад фрагменту вмісту файлу out_res_log

res array	norm probs	prob array
150477	0.8	0.000953547
132159	0.76591	0.000912917
113848	0.7359	0.00087714
96160	0.66613	0.000793981
84342	0.56658	0.000675329
77424	0.53459	0.00063719
69144	0.48938	0.000583309
60003	0.46188	0.000550531
51383	0.40974	0.000488388
41831	0.3695	0.000440418
33512	0.31755	0.000378496
25516	0.28742	0.000342582
19476	0.24629	0.000293565
13969	0.22464	0.000267754
10447	0.16399	0.000195461
7278	0.13472	0.000160572
5214	0.06306	7.5164E-05
3635	0.03719	4.43251E-05
2490	0.01823	2.17339E-05
1697	0.00786	9.3744E-06

Рисунок. 3.3 - Приклад фрагменту вмісту файлу res_log

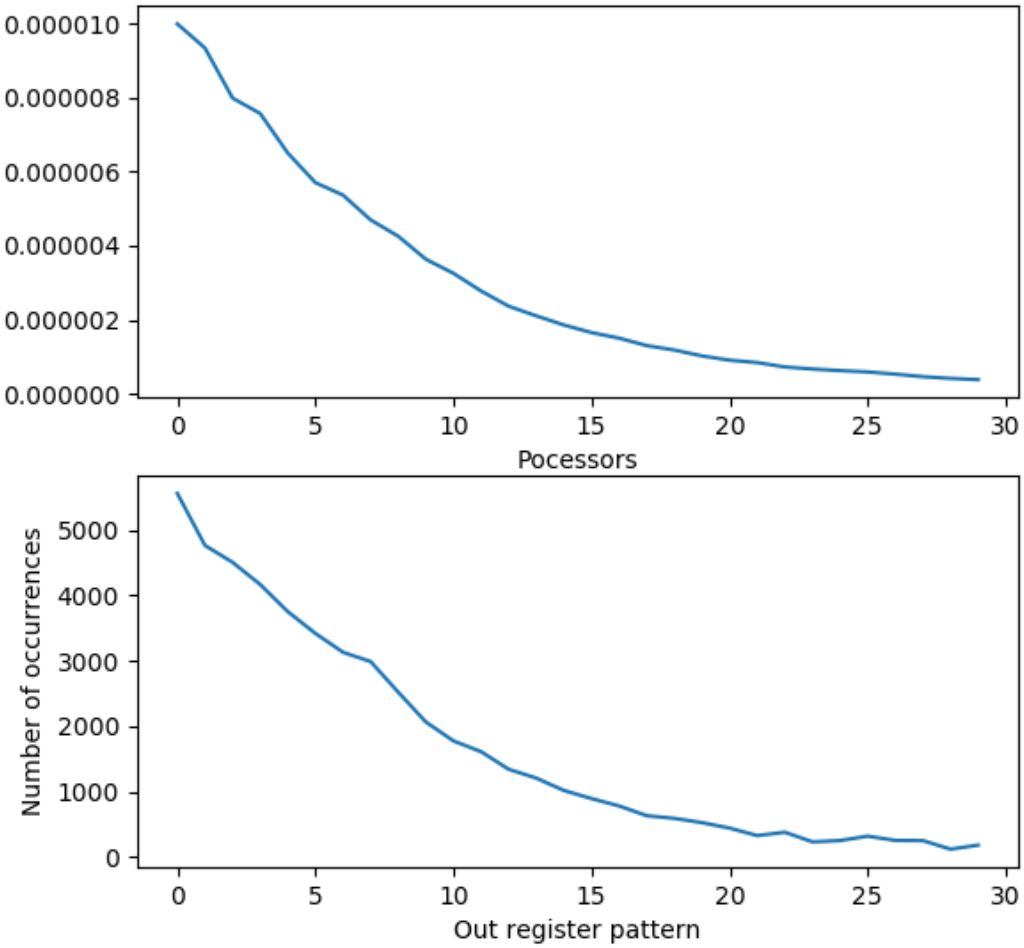


Рисунок 3.4 - Приклад графіків результату генерації

Розроблена програмна модель дозволяє провести низку тетів з різними вхідними параметрами, що, в свою чергу, дозволяє проаналізувати зміни в результатах тестів та краще ознайомитися зі впливом змінюваних параметрів на роботу моделі. Також розроблений програмний продут дозволяє моделювати роботу генератора з рівноймовірним розподілом одиничних значень у вихідних векторах. Завдяки цьому можна провести порівняння роботи рівноймовірного генератора та генератора, створеного за допомогою нового розробленого метода.

Наприклад, частота проведення взаємозаміни крайніх розрядів сильно впливає на остаточний розподіл, як можна побачити з рисунку 3.5 нижче.

Рисунок 3.5 - Результати роботи теста 1 та теста 2 з різною частотою

norm probs	prob array	N = 3	N = 5
0.8	0.000953547	124987	101421
0.79947	0.000952917	108441	84982
0.77784	0.00092714	92065	71206
0.58223	0.000693981	70923	58761
0.48269	0.000575329	67284	57312
0.47586	0.00056719	65452	56219
0.46421	0.000553309	63076	55553
0.43671	0.000520531	59601	54383
0.40974	0.000488388	56265	52930
0.36111	0.000430418	51818	51078
0.25043	0.000298496	47127	49523
0.2203	0.000262582	41736	47132
0.21273	0.000253565	35554	44570
0.20786	0.000247754	29440	41488
0.16399	0.000195461	24138	38208
0.15988	0.000190572	19122	34333
0.06306	7.5164E-05	15429	30920
0.03719	4.43251E-05	11904	27160
0.02662	3.17339E-05	8901	23265
0.01153	1.3744E-05	6742	19561

взаємозаміни розрядів

Різниця між тестом з частотою взаємозаміни розрядів з кожен 3 такт та тестом з частотою заміни кожен 5 такт становить від близько -20 відсотків до близько +250 відсотків різниці у кількості появи одиничних значень в

залежності від номера розряду.

Також на вигляд остаточного розподілу досить сильно впливають параметри, що відповідають за вибір розрядів, значення яких буде взаємозамінюватися. Так, при однакових вхідних ймовірностях, тест з параметрами sw_pos_pr1 = 50 та sw_pos_pr0 = 55 сильно відрізняється від аналогічного тесту з параметрами sw_pos_pr1 = 80 та sw_pos_pr0 = 85. Цю різницю можна побачити на рисунку 3.6, рисунку 3.7 та рисунку 3.8, що представлені нижче.

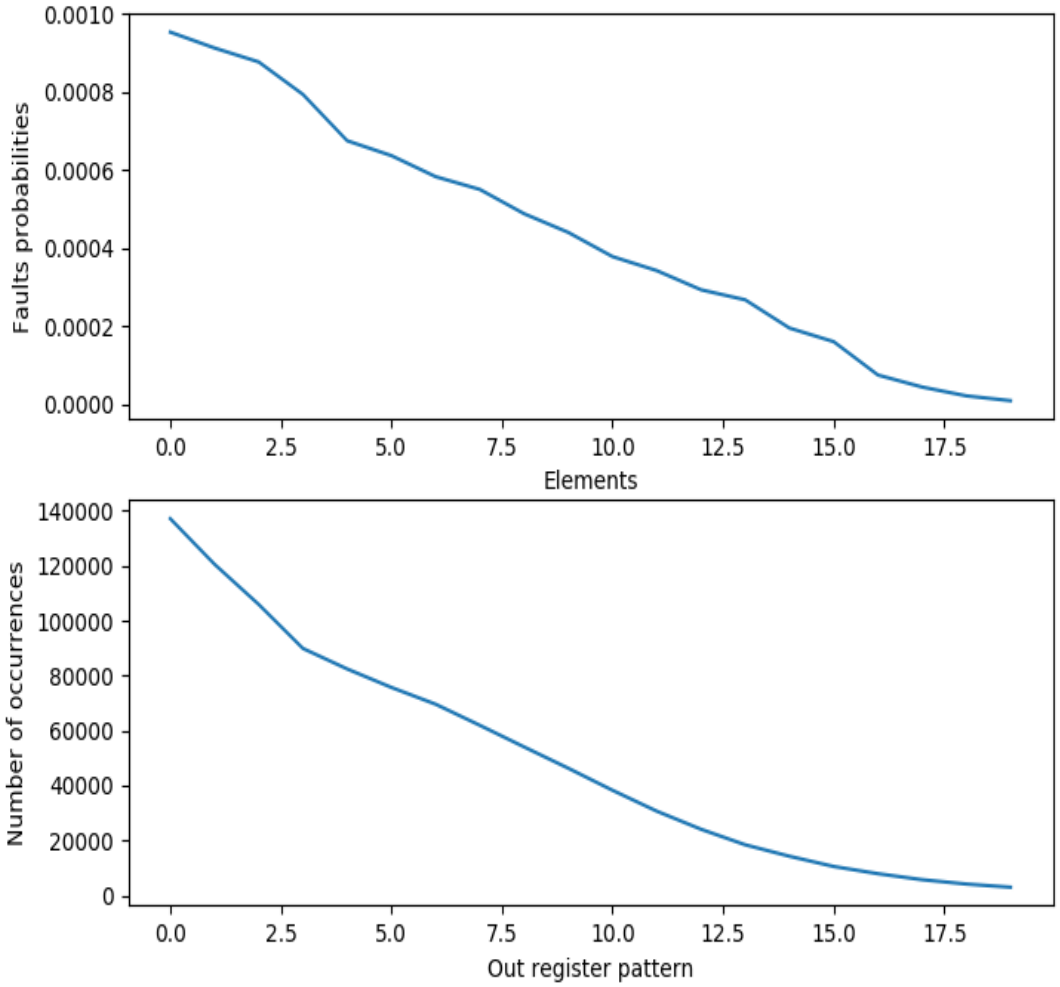


Рисунок 3.6 - Графік результатів теста 1, p1 = 50, p2 = 55

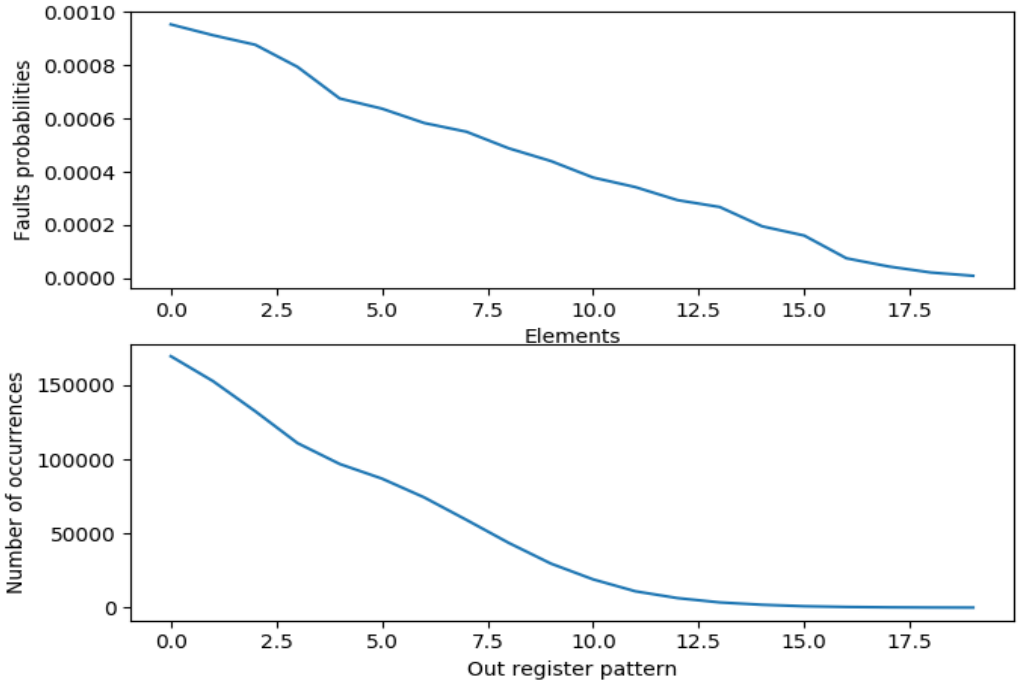


Рисунок 3.7 - Графік результатів теста 2,

norm probs	prob array	p1=50, p2=55	p1=80, p2 = 85
0.8	0.000953547	137156	169467
0.76591	0.000912917	120567	152641
0.7359	0.00087714	105917	132358
0.66613	0.000793981	89885	110911
0.56658	0.000675329	82420	96818
0.53459	0.00063719	75694	86973
0.48938	0.000583309	69583	74312
0.46188	0.000550531	61893	59181
0.40974	0.000488388	54030	43694
0.3695	0.000440418	46261	29716
0.31755	0.000378496	38234	19077
0.28742	0.000342582	30631	11053
0.24629	0.000293565	24050	6451
0.22464	0.000267754	18392	3545
0.16399	0.000195461	14217	1960
0.13472	0.000160572	10494	971
0.06306	7.5164E-05	7869	483
0.03719	4.43251E-05	5678	232
0.01823	2.17339E-05	4094	110
0.00786	9.3744E-06	2940	52

Рисунок 3.8 - Результати роботи теста 1 та теста 2 з різними параметрами вибору розрядів для взаємозаміни

Як видно з рисунків вище, різниця в кількості появи одиничного значення між двома даними тестами становить від близько +25 відсотків до близько -97 відсотків.

Також, використовуючи розроблену модель, можливо порівняти роботу генератора зі зміненим параметром налаштовування функцій збудження, що є однією з особливостей розробленого методу.

Тобто в першому тесті ймовірності функцій збудження будуть залежати від вхідних ймовірностей, а в іншому вони всі будуть дорівнювати 0,5. Результат проілюстровано на рисунку 3.9.

norm probs	prob array	pi/p1	test1	test1 ci/c1	test 2	test2 ci/c1
0.65	0.000953547	100	103387	100	87211	100
0.622304	0.000912917	95.73907692	94566	91.46797953	87685	100.5435094
0.597916	0.00087714	91.98707692	89908	86.9625775	90219	103.4491062
0.541229	0.000793981	83.266	86232	83.40700475	90654	103.9478965
0.460348	0.000675329	70.82276923	84359	81.59536499	88586	101.576636
0.434351	0.00063719	66.82323077	81225	78.5640361	85573	98.12179656
0.397621	0.000583309	61.17246154	77135	74.60802615	79621	91.29696942
0.375278	0.000550531	57.73507692	70889	68.56664764	72798	83.47341505
0.332917	0.000488388	51.218	63336	61.26108698	63668	73.00455218
0.300218	0.000440418	46.18738462	54816	53.02020564	54665	62.68131314
0.258007	0.000378496	39.69338462	46448	44.9263447	45927	52.66193485
0.233527	0.000342582	35.92723077	37891	36.64967549	37246	42.70791529
0.200113	0.000293565	30.78661538	30054	29.06941879	29761	34.12528236
0.182519	0.000267754	28.07984615	23266	22.50379642	23283	26.69732029
0.133239	0.000195461	20.49830769	17721	17.14045286	18125	20.78292876
0.109456	0.000160572	16.83938462	13240	12.80625224	13897	15.93491647
0.0512367	7.5164E-05	7.882569231	9793	9.472177353	10904	12.50300994
0.0302149	4.43251E-05	4.648446154	7104	6.871270082	8506	9.753356801
0.0148152	2.17339E-05	2.279261538	5066	4.900035788	6645	7.619451675
0.0063902	9.3744E-06	0.983107692	3569	3.452078114	5031	5.768767701

Рисунок 3.9 - Результати тесту 1 та тесту 2 з різними налаштуваннями функцій збудження

З даних результатів можна побачити, з якими налаштуваннями функцій збудження результуючий розподіл буде більше схожий на вхідний розподіл ймовірностей. Якщо порівняти відношення нормалізованих значень вхідних ймовірностей відмов до максимального нормалізованого значення ймовірності відмови до відношення значень кількості появ одиничного значення у розрядах до кількості появи одиниць у першому розряді у тесті 1 та тесті 2, то в середньому, відмінність у відношеннях ймовірностей і

кількості появи одиничного значення в тесті 1 буде становити менше 10 відсотків, у той час коли в тесті 2 ця різниця буде більше 25 відсотків.

Проте одним з найголовніших порівнянь є порівняння роботи нового керованого генератора та звичайного рівноймовірного генератора. Результати тесту з керованим і звичайним генераторами представлені нижче на рисунках 3.10, 3.11, та 3.12.

norm probs	prob array	test 1	test 2
0.65	0.000953547	104212	49447
0.622304	0.000912917	94556	50110
0.597916	0.00087714	90617	50013
0.541229	0.000793981	86643	50405
0.460348	0.000675329	83563	50247
0.434351	0.00063719	81486	49774
0.397621	0.000583309	76707	49970
0.375278	0.000550531	70990	50352
0.332917	0.000488388	63632	50554
0.300218	0.000440418	54939	50002
0.258007	0.000378496	46121	50106
0.233527	0.000342582	37658	49941
0.200113	0.000293565	29715	50325
0.182519	0.000267754	22971	49613
0.133239	0.000195461	17457	49279
0.109456	0.000160572	13136	49802
0.0512367	7.5164E-05	9764	50204
0.0302149	4.43251E-05	7124	50318
0.0148152	2.17339E-05	5107	49437
0.0063902	9.3744E-06	3607	50106

Рисунок 3.10 - Результати тестів з керованим генеатором та рівноймовірним генератором

З результатів легко побачити, що в результаті роботи моделі рівноймовірного генератота було згенеровано послідовность векторів з майже однаковою кількістю одиничних значень. Теоритично, ця кількість повинна дорівнювати $C = k/n * m$ де C - кількість одиничних значень, k - вага вектору, тобто кість одиничних значень у векторі, n - довжина вектору, m - кількість згенерованих векторів. У кожному тесті було згенеровано 200000 векторів, таким чином розрахункова кількість одиничних значень у розряді дорівнює 50000, що і було отримано в результаті. В свою чергу розподіл керованого генератора в середньому відрізняється від вхідного розподілу ймовірностей відмов менш ніж на 10 відсотків.

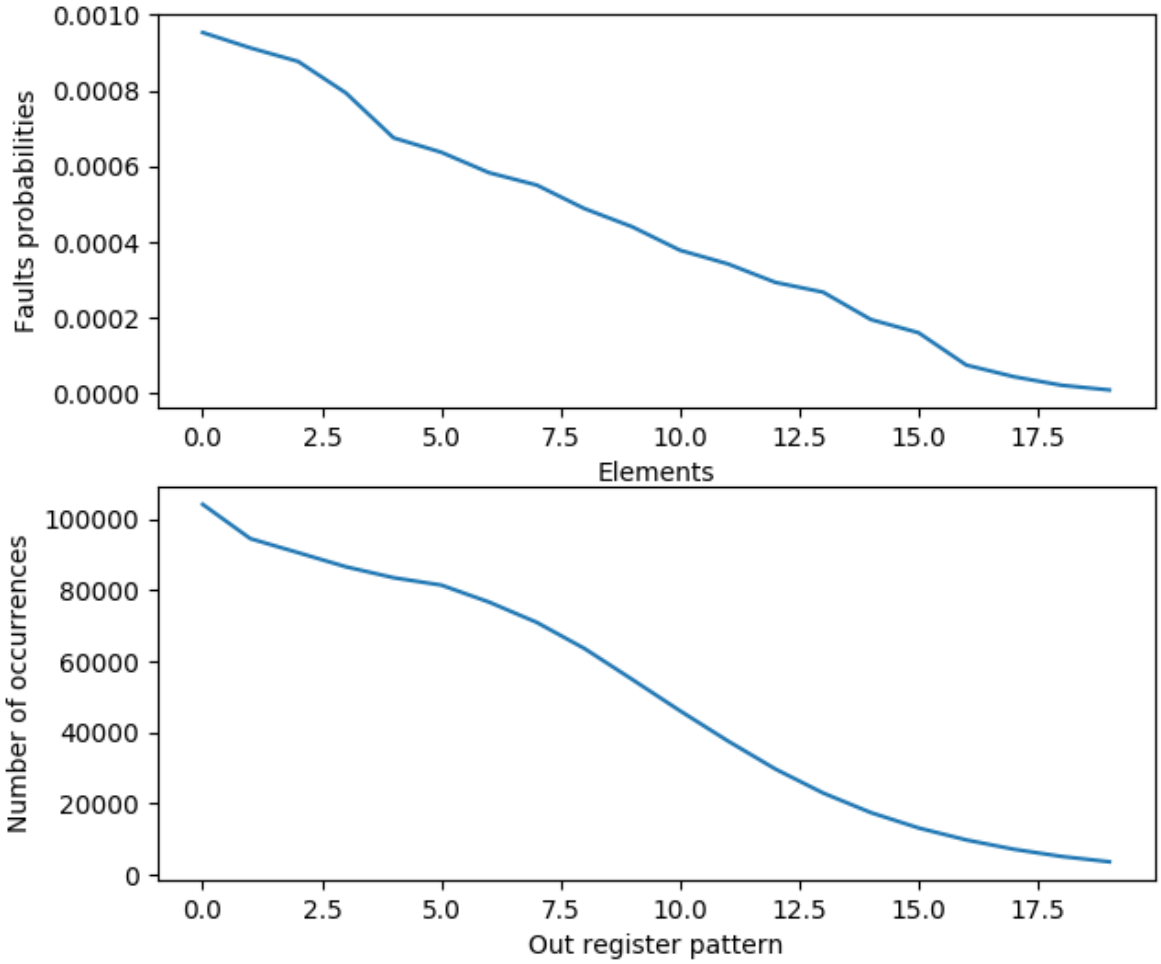


Рисунок 3.11 - Графік результатів теста 1 з керованим генератором

Для опису різниці між результатами рівноймовірного генератора та керованого генератора можна описати певну функцію ваги W, таку, що $W = \sum_{i=1}^{m} w_i n_i$, де w_i - елемент вхідного вектору ймовірностей відмов, нормалізований так, що $w_1 = p_i/p_1$, n_i = кількість появи одичного значення у відповідному розряді вихідних векторів. Розрахувавши функцію W для тесту з керованим генератором та рівноймовірним генератором, отримаємо, що $W_{кер}$ для керованого генератора більше за значення $W_{рів}$ для рівноймовірного генератора більш ніж на 40 відсотків.

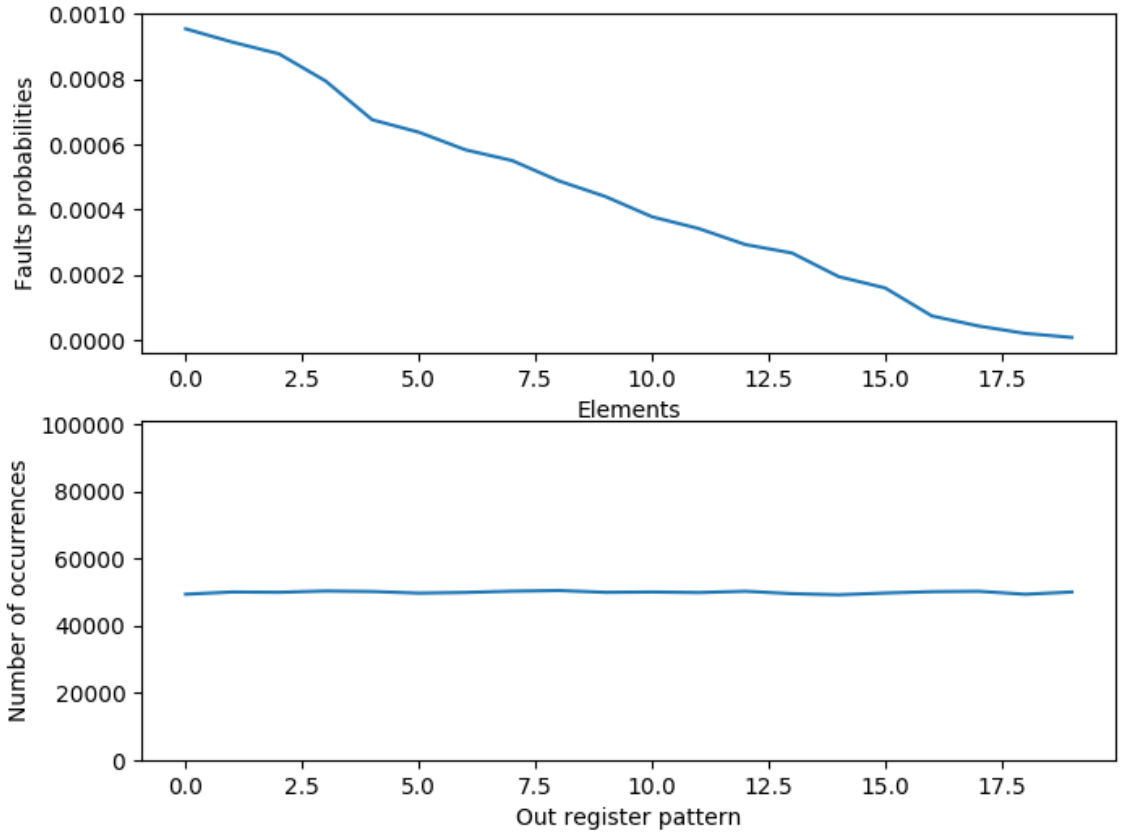


Рисунок 3.12 - Графік результату теста 2 з рівноймовірним генератором

3.5. Висновки

В ході роботи було розроблено новий структурний метод синтезу генератора послідовності двійкових векторів заданої ваги з різною ймовірністю появи наборів, які залежить від вхідних даних, введених користувачем та створено програмний продукт, що дає змогу моделювати роботу такого генератора, а також візуалізувати результати генерації.

Послідовності і графіки результатів можуть бути побудовані для довільних значень довжини генерованих векторів, ваг генерованих векторів та вхідних ймовірнісних векторів. Графіки дозволяють проаналізувати якість розподілу значень сформованих векторів відносно вектору вхідних послідовностей.

ВИСНОВКИ

В даній роботі проведено аналіз існуючих методів генерації псевдовипадкових чисел та псевдовипадкових тестових послідовностей спеціального виду, розглянуто питання тестування цифрових систем використовуючи моделі їх поведінки в потоці відмов.

Розроблено метод синтезу керованого генератора псевдовипадкових двійкових векторів постійної ваги, що базується на існуючому методі синтезу генератора двійкових векторів постійної ваги. На базі цього нового методу розроблено програмну модель генератора, що дозволяє проводити статистичні експерименти з моделями неоднорідних цифрових систем.

Новий метод дозволяє створювати генератори послідовності двійкових векторів, розподіл одиничних значень яких залежить від розподілу вхідних ймовірностей відмов, але вага векторів залишається постійною. З аналізу результатів роботи моделі можна побачити, що розроблений метод дозволяє оптимізувати процес тестування з моделями поведінки неоднорідних цифрових систем. Це досягається за рахунок того, що за одиницю часу буде згенеровано послідовність векторів з більшу кількість одиничних значень на позиціях, що відповідають елементам з більшою ймовірністю відмови, що мають більший пріоритет у процесі розрахунку надійності системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Хамитов Г. П., Имитация случайных процессов. // Иркутск, Изд-во Иркут. института, 1983. -184 с.
2. Бухштаб А. А. Теория чисел.-М, Просвещение, 1966. // 384 с.
3. Голенко Д. И., Моделирование и статистический анализ псевдослучайных чисел на электронных машинах. // М., Наука, 1965 -227 с.
4. Kashivagi H., Recurrence of M-type Sequences. // J. Soc. Instrum. and Control Eng. (Japan), 1981, vol. 20, N 2, p. 236-245
5. В. Н. Ярмолик, С. Н. Демиденко. Генерирование и применение псевдослучайных сигналов в системах испытаний и контроля. // Минск, Наука и техника, 1986 - 472с
6. Yarmolik V. N., Method of Pseudorandom Sequence Formation (M-sequences) for Computational Statistics. // 6-th Symposium on Computational Statistics. 1984. CSSR, Prague, p. 121.
7. Бернштейн М. С., Романкевич А. М., Метод статистического контроля логических схем. // Кибернетика, 1974, №4, с. 52-67.
8. Lewis T. G., Payne W. H. Generalized Feedback Shift Register Pseudorandom Number Algorithm. // Journal of ACM, 1973, vol. 20, N 3, p. 456-468
9. В. О. Романкевич, І. В. Майданюк Структурний методу формування двійкових псевдовипадкових векторів заданої ваги. // Вісник НТУУ ”КПІ”. Інформатика, управління та ОТ. -2009- Вип. 49. –с.34-42